

School of Computer Applications

Project Submission Form

Lecturer: Renaat Verbruggan
Project Title: The Object Oriented Project

Course Title: Software Engineering
Course Code: CA
Degree Programme: B.Sc.
Stage: 4

Student Name: Mark Cunningham
Student Number: 95309021
Date:

Declaration

I the undersigned declare that the project material, which I now submit, is my own work. Any assistance received by way of borrowing from the work of others has been cited and acknowledged within the text of this work. I make this declaration in the knowledge that a breach of the rules pertaining to project submission may carry serious consequences.

Signed

The Object Oriented Project

The goal is to use the features of an object oriented language correctly and appropriately.

Description of Project

This application, written in JAVA, is based on the infamous Tamagotchis, the virtual pet that beeps when it's hungry or sick and the owner must hit the appropriate button to feed or play with it.

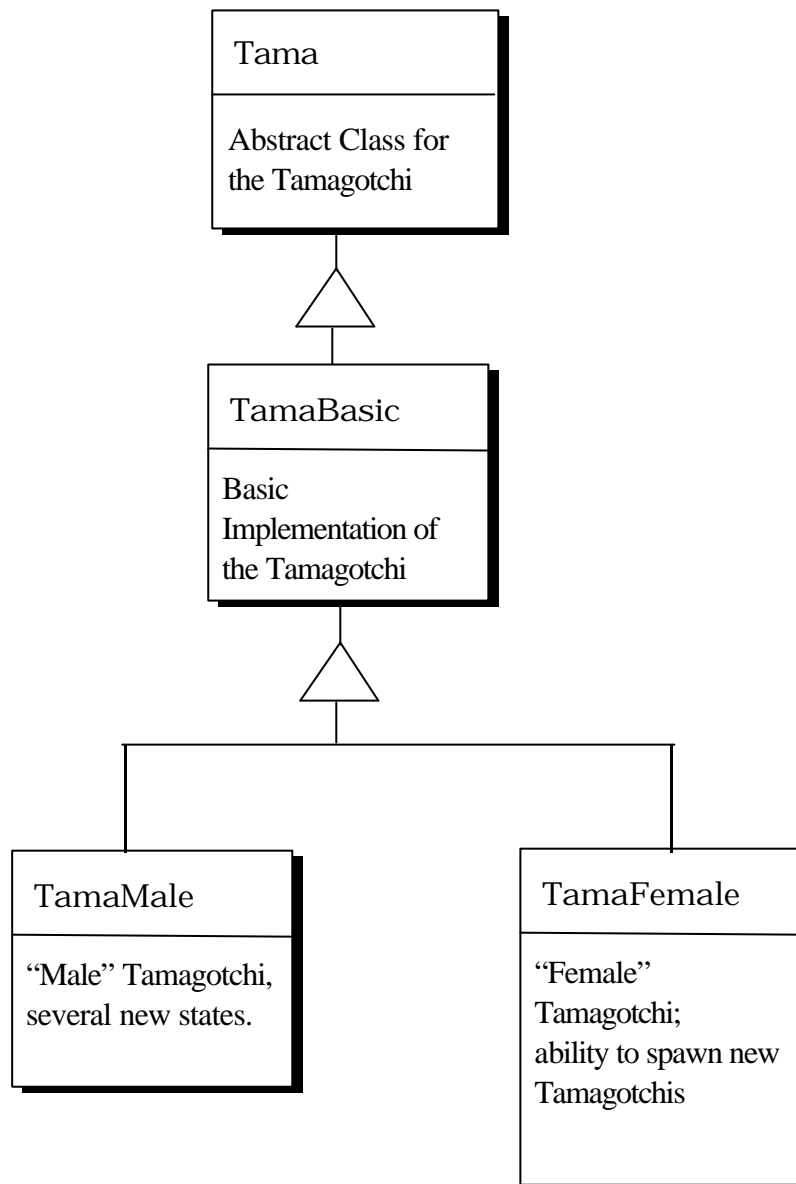
To demonstrate the idea of Object-Oriented, the idea is extended to be a "family" of Tamagotchis. A basic tamagotchi with several other types that extend the original. The idea is that you can have several tamagotchis running and they would interact, breed and multiply.

The Design Description

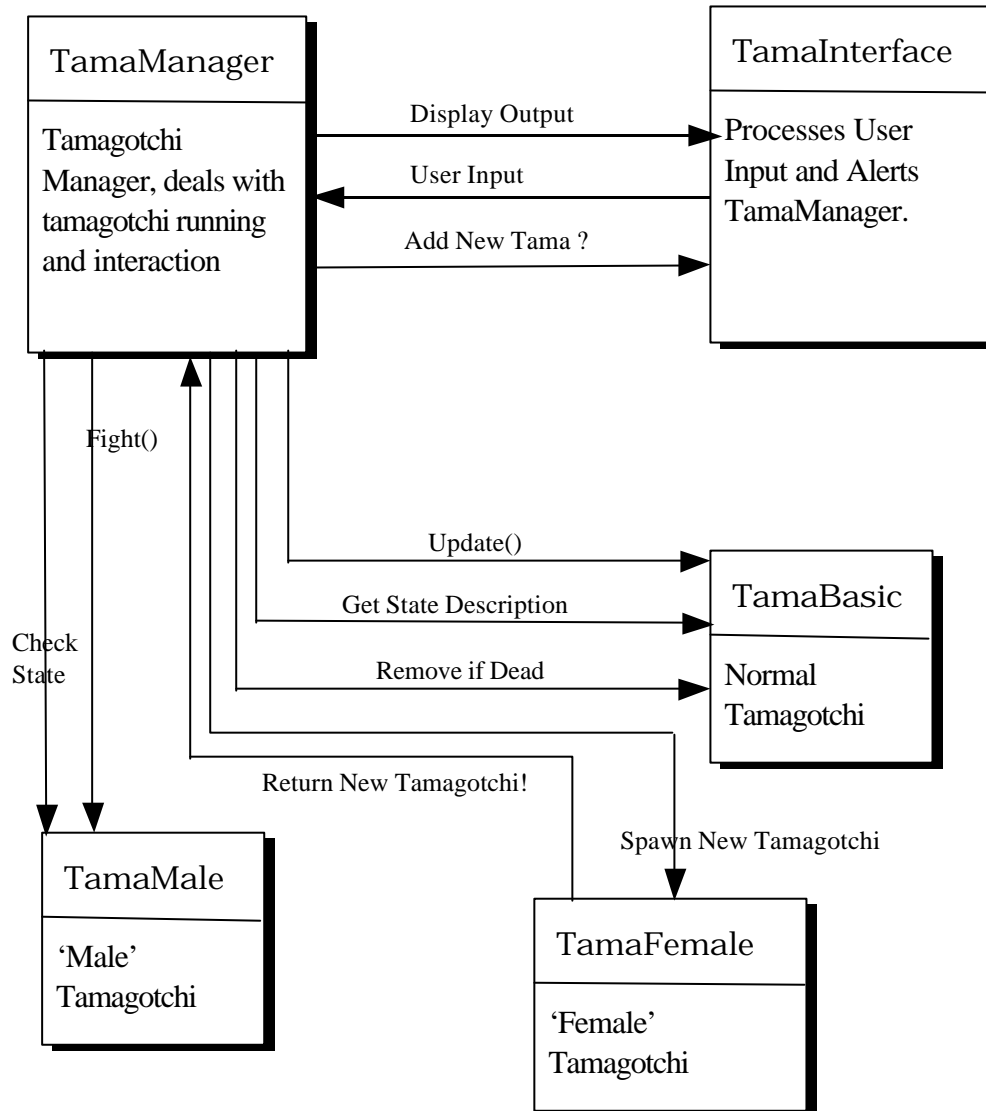
The main class is the Tamagotchi set of classes. Four Classes, a abstract class Tama and then a TamaBasic which is a basic implementation of the abstract class. Then a TamaMale and TamaFemale who are both inherited from TamaBasic, extending the normal tamagotchi to allow breeding and fighting.

The TamaBasic holds several variables (_tiredness, _health etc.) that vary due to time and interaction. These variables define what state the TamaBasic is in (HUNGRY, BORED, ASLEEP etc.). TamaMale extends the TamaBasic to allow extra states and inter Tamagotchi interaction such as FIGHTING. The TamaFemale also extends TamaBasic allows the ability to "breed", i.e. spawn a new Tamagotchi. The design also calls for a Manager that updates and deals with responses from Tamagotchis. Refer to the code listings for more detailed description of states, variables and implementation.

The Class Hierarchy



Class Interaction



Code Listings

Some notes on code:

All Code presented here compiles and works is without any major bugs. Not included here is the testing classes used during development.

It was developed in JAVA using the JDK 1.1.6 and JDK 1.1.5 under NT and Win95. Programmer's File Editor (32 bit) was used to edit files.

There is 7 separate files; Tama.java, TamaBasic.java, TamaFemale.java, TamaMale.java, TamaManager.java, TamaInterface.java and TamaApplication.java.

The debugging output is left in, so that some output can be generated and presented.

Tama.java

```
////////////////////////////////////  
/  
//  
// File: Tama.java  
//  
// Written by Mark Cunningham 95309021  
// unless otherwise stated.  
//  
// Compiler: javac; JDK 1.1.6 (and JDK 1.1.5)  
// Operating System used: Windows 95 and Windows NT  
//  
// From today 4th December 1998 everything complies  
// and bugs as reported in code  
//  
////////////////////////////////////  
/  
  
import java.lang.*; /* basic java classes inc Object */  
  
////////////////////////////////////  
/
```

```
//
// This is the most basic class of the hierarchy
// These functions defines the interface to any Tamagotchi
//
////////////////////////////////////
/

abstract class Tama extends Object {

/* updates the current state of the
   tamagotchi */
abstract public void update();

/* the basic functions of the Tamagotchi,
to feed, to play, to sleep and to kill */
abstract public void Feed();
abstract public void Sleep();
abstract public void Play();
abstract public void Kill();

}
```

TamaBasic.java

```
////////////////////////////////////
/
//
// File: TamaBasic.java
//
// Written by Mark Cunningham 95309021
// unless otherwise stated.
//
// Compiler:          javac; JDK 1.1.6 (and JDK 1.1.5)
// Operating System used:  Windows 95 and Windows NT
//
// From today 4th December 1998 everything complies
// and bugs as reported in code
//
////////////////////////////////////
/

import java.lang.*;
import Tama;

////////////////////////////////////
//
//
// This is the basic implementation of the abstract class Tama
// TamaFemale and TamaMale are derived from this class
//
// TamaBasic can be in several states OK, BORED, HUNGRY, TIRED,
ASLEEP
// and DEAD. These states depend on the values of constantly
changing
// variables, _health, _bored and _tiredness.
//
// ASLEEP is a special state that allows regeneration of certain
// variables.
//
// DEAD marks the TamaBasic as finished and whatever Manager
// class is running it must clean up.
//
////////////////////////////////////
/

public class TamaBasic extends Tama {

/* these variables control the behaviour of the Tamagothchi */
protected int _health;      /* health is how 'healthy' the tama is
-
                                if health <= 0 tama is dead */
protected int _age;        /* keeps track of age of tama */
protected int _bored;     /* how entertained is tama, the higher
the more bored */
protected int _tiredness; /* how tired the tama is */
```

```

protected int _tolerance;    /* how much to increase or decrease
things */
protected int _state;      /* current state of Tama */

/* these are the max values of the behaviour variables */
protected int max_health = 20;
protected int max_bored = 20;
protected int max_tiredness = 20;

/* static final variables corresponding to the 6 states
order defines priority- Dead is way more important
then Okay */
protected static final byte OK = 0;    /* everything is fine */
protected static final byte BORED = 1; /* bored - nothing to do
*/
protected static final byte HUNGRY = 2; /* hungry */
protected static final byte TIRED = 3; /* exhausted */
protected static final byte ASLEEP = 4; /* sleeping- regaining
health */
protected static final byte DEAD = 5;  /* tama is dead - do
nothing */

/* miscellenous information */
protected String _name = "Undefined";
protected String _type = "Undefined";

/* class constructor */
public TamaBasic(int h, int b, int t, int tol, String n){
    setState(OK);
    setAge(0);
    setHealth(h);
    setBored(b);
    setTiredness(t);
    setTolerance(tol);
    setName(n);
    _type = "BASIC";
}

/*
=====
implement abstract methods from Tama class
=====
*/

/* update states and variables */
public void update(){

    /* intial checks */
    if(_health <= 0 ){ setState(DEAD); }
    else { setAge(getAge()+1); }

    /* check for special states (ASLEEP and DEAD) */

    if(_state == ASLEEP){
        stateAsleep();

```

```

    }
    else if(_state == DEAD){
        stateDead();
    }
    else {

        /* natural decrease of health, boredom and increase of
tiredness */
        _health--;
        _bored++;
        _tiredness++;

        /* check state and weather time to change state */

        stateOk();

        if(_bored > ((max_bored / 4) * 3)){
            stateBored();
        }

        if(_tiredness > ((max_tiredness / 4) * 3)){
            stateTired();
        }

        if(_health < (max_health / 2)){
            stateHungry();
        }

    }

}

/* interactive functions to users */

/* if tama is hungry :- feed */
public void Feed() { if(getState() != ASLEEP) incHealth(); }

/* tama is bored:- entertain it */
public void Play() { if(getState() != ASLEEP) decBored(); }

/* tama is tired:- make it sleep */
public void Sleep(){ setState(ASLEEP); }

/* Remove tama : kill */
public void Kill(){
    if(getState() != DEAD){
        setState(DEAD);
        setHealth(-10);
    }
}

/*
=====
some internal functions for the working of the tamagotchi
=====

```

```

*/

/* state functions - can only be in one state at one time */
protected void stateBored(){

    /* state bored is due to high boredom, the lowest priority state
       after ok, play with it and it should be fine */

    setState(BORED);

}
protected void stateHungry(){

    /* state hungry is due to low health - feed it */

    setState(HUNGRY);

}
protected void stateTired(){

    /* state tired is due to high tiredness - can lead to decrease
       in health:- sleep */

    setState(TIRED);

}
protected void stateAsleep(){

    /* sleep is a special user state. If high tiredness, relaxs it.
       if high boredom, relaxs it. */

    /* if low tiredness, incese boredom, decrease health */

    /* sleep is a factor of tiredness */

    /* must test these functions - could be possible bug! */

    if(_tiredness < (max_tiredness / 4)){
        setState(OK);
        _bored++;
        _health--;
    }
    else {
        setState(ASLEEP);
        _tiredness--;
        _bored--;
        _health++;
    }

}
protected void stateOk(){

    /* the magical mystically state of perfection and golconda */
    /* everything is balanced though there is a natural tendency

```

```

        for health to decrease and boredom and tiredness to increase
*/

    setState(OK);

}
protected void stateDead(){

    /* erm. if tama is in this state... it's a dead weight... Tama
manager
    should remove it shortly */

    setState(DEAD);

}

/* increase appropite variable (internal interface functions) */
protected void incHealth(){

    /* if health is high as in the top 25% increase tiredness and
boredom */
    /* if very health in fact decrease health */
    /* if low health (as in bottom 50%) increase health */

    if (_health > ((max_health / 4) * 3)){
        _bored++;
        incTiredness();

        if(_health > ((max_health / 100) * 90)){
            _health-=_tolerance;
        }
    }
    else if(_health <= (max_health / 2)){
        _health+=_tolerance;
    }
}

protected void decBored(){

    /* if high boredom (top 25%) decrease boredom */
    /* if low boredom (bottom 50%) increase boredom and tiredness */

    if(_bored > ((max_bored / 4) * 3)){
        _bored-=_tolerance;
    }
    else if(_bored <= (max_bored / 2)){
        _bored+=_tolerance;
        incTiredness();
    }
}

protected void incTiredness(){

    /* if high tiredness (top 25%) decrease health */

```

```

        if(_tiredness > ((max_tiredness / 4) * 3)){
            _health-=_tolerance;
        }
    }

    /*
    =====
    the rest of the public access functions
    to allow customising of the tamagotchi
    =====
    */

    /* accessor functions */
    public String getType() { return _type; }
    public int getHealth() { return _health; }
    public int getAge() { return _age; }
    public int getBored() { return _bored; }
    public int getTiredness(){ return _tiredness; }
    public String getName() { return _name; }
    public String getStateDescpt(){

        String ret_string = "no state defined";

        if (_state == BORED){
            ret_string = getName() + " is Bored...";
        }
        else if(_state == HUNGRY){
            ret_string = getName() + " is Hungry...";
        }
        else if(_state == TIRED){
            ret_string = getName() + " is Sleepy...";
        }
        else if(_state == ASLEEP){
            ret_string = getName() + " is Sleeping...";
        }
        else if(_state == DEAD){
            ret_string = getName() + " is Dead.";
        }
        else if(_state == OK){
            ret_string = getName() + " is Okay!";
        }
        else {
            ret_string = new String("Error: Invalid State.");
        }

        return ret_string;
    }
    protected int getState(){ return _state; }

    /* set functions:- should only be used by the Manager class,
    debug and constructor functions and internal functions */
    public void setHealth(int h) { _health = h; }

```

```

public void setBored(int b)           { _bored = b;           }
public void setTiredness(int t)      { _tiredness = t;      }
public void setTolerance(int t)      { _tolerance = t;      }
public void setMaxHealth(int mh)     { max_health = mh;     }
public void setMaxTiredness(int mt)  { max_tiredness = mt;  }
public void setMaxBored(int mb)      { max_bored = mb;      }
public void setName(String newname)  { _name = newname;    }
/* protected- should only be called within class */
protected void setAge(int a)         { _age = a;           }
protected void setState(int s){
    /* only change to valid states */
    if(s >= 0 && s <= 5){
        _state = s;
    }
}
}
}

```

TamaFemale.java

```
////////////////////////////////////
/
//
// File: TamaFemale.java
//
// Written by Mark Cunningham 95309021
// unless otherwise stated.
//
// Compiler:          javac; JDK 1.1.6 (and JDK 1.1.5)
// Operating System used:  Windows 95 and Windows NT
//
// From today 4th December 1998 everything complies
// and bugs as reported in code
//
////////////////////////////////////
/

import TamaBasic; /* the basic tamagotchi */
import TamaMale;  /* the male tamagotchi */

////////////////////////////////////
/
//
// this class extends the TamaBasic class and adds certain extras.
//
// It can "give birth" to a new Tamagotchi. When the Breed
function
// is called, as long as Tamagotchi is awake (not ASLEEP) and
alive
// (not DEAD) it will randomly spawn a new Tamagotchi and return
it
// to the calling class, assuming some sort of Manager class.
//
// It can only breed twice, the first time will half it's health,
// the second time will kill it.
//
////////////////////////////////////
/

public class TamaFemale extends TamaBasic {

private int hasBreed; /* has the tamagotchi breed or at least
attempted to? */

/* constructors */
public TamaFemale(int h, int b, int t, int tol, String n){

    super(h,b,t,tol,n);
    hasBreed = 0;
    _type = "FEMALE";
}
}
```

```

/* female Tamagotchi around - can breed - returns a new tamagotchi
*/
public TamaBasic Breed(){

    TamaBasic baby = null;
    double randomType = Math.random();

    /* as long as alive and not asleep */
    if(_state != ASLEEP && _state != DEAD){

        /* randomly select a new Tamagotchi to spawn, TamaMale,
TamaFemale and TamaBasic */
        if(randomType < 0.3){ baby = new TamaBasic(max_health, 0, 0,
_tolerance, "Baby"); }
        else if(randomType < 0.6){ baby = new TamaFemale(max_health,
0, 0, _tolerance, "Baby Girl"); }
        else if(randomType <= 1){ baby = new TamaMale(max_health, 0,
0, _tolerance, "Baby Boy"); }

        /* only breeds twice - first time halves health, second time
kills */
        hasBreed++;
        if(hasBreed == 1){ setHealth(_health/2); }
        else if(hasBreed >= 2){ setState(DEAD); }
    }

    return baby; /* return pointer to new Tamagotchi */
}

/*
=====
Overwritten functions
=====
*/

/* changed descriptions to allow 'personality' from
the users perspective
among different tamagotchis
*/
public String getStateDescpt(){

    String ret_string = "no state defined";

    if (_state == BORED){
        ret_string = getName() + " is in need of passion.";
    }
    else if(_state == HUNGRY){
        ret_string = getName() + " wants her chocolate and ice-
cream.";
    }
    else if(_state == TIRED){
        ret_string = getName() + " wants her warm bed.";
    }
    else if(_state == ASLEEP){

```

```
        ret_string = getName() + " is fast asleep!.";
    }
    else if(_state == DEAD){
        ret_string = getName() + " died. :(";
    }
    else if(_state == OK){
        ret_string = getName() + " is shopping. :)";
    }
    else {
        /* error state */
        ret_string = new String("Error: Invalid State.");
    }

    return ret_string;
}
}
```

TamaMale.java

```
////////////////////////////////////
/
//
// File: TamaMale.java
//
// Written by Mark Cunningham 95309021
// unless otherwise stated.
//
// Compiler:          javac; JDK 1.1.6 (and JDK 1.1.5)
// Operating System used:  Windows 95 and Windows NT
//
// From today 4th December 1998 everything complies
// and bugs as reported in code
//
////////////////////////////////////
/

import TamaBasic;

////////////////////////////////////
/////
//
// this class extends the TamaBasic class and adds certain extras.
//
// it is a virtual "male" tamagotchi. It adds two new states,
BREEDING and
// FIGHTING. When it has a certain high state it will wish to
breed. If it
// isn't told it has breed it will get angry and wish to fight.
//
// The TamaMale takes a TamaBasic into it's fight function and
works
// out the winner dependant on health, tiredness and age. Age is
the more
// powerful variable here.
//
////////////////////////////////////
/////

public class TamaMale extends TamaBasic {

    /* new states fighting where male is wishing to fight */
    protected static final byte FIGHTING = 6;
    /* breeding wishes to breed, which can be quite often */
    protected static final byte BREEDING = 7;

    private boolean hasBreed; /* has the tamagothic breed or at least
    attempted to? */
    private int previousState; /* memory of old state */

    /* constructors */
}
```

```

public TamaMale(int h, int b, int t, int tol, String n){

    super(h,b,t,tol,n);
    hasBreed = false;
    previousState = getState();
    _type = "MALE";
}

/*
=====
Extensions to TamaBasic
=====
*/

/* update is same as before but 2 new states are added */
public void update(){

    super.update(); /* call super class update */

    /* as long as valid */
    if(_state != DEAD && _state != ASLEEP){

        /* if content and high health it will attempt to breed -
        but if it can't it will fight (ie kill other Tamagotchis)
        must be a little old */

        if(previousState == BREEDING && hasBreed == false){
stateFight(); }

        if(previousState != BREEDING
        && previousState != FIGHTING
        && (_health > (max_health / 2)) /* good health */
        && (_bored < (max_bored / 4)) /* very content */
        && (_age > ((max_health / 4) * 3))){ /* at least some
maturity */
            hasBreed = false;
            stateBreed();
        }

    }

    previousState = getState(); /* update memory of last state for
next loop */
}

/* new state functions */
protected void stateBreed(){ setState(BREEDING); }
protected void stateFight(){ setState(FIGHTING); }

/* Tamagothic around - can breed */
public void Breed(){ hasBreed = true; }
public void Fight( TamaBasic victim ){ /* Breed(); currently has
the same effect */

```

```

int victimState = victim.getState();
int maleState = this.getState();
int victimFight = 0, maleFight = 0;

/* as long as a valid victim */
if(victimState != DEAD && maleState == FIGHTING){

    /* if victim ASLEEP, easy target */
    if (victimState == ASLEEP){
        victim.Kill();
        Breed();
    }
    else { /* fighting - factor of health, tiredness and age
(experince) */

        victimFight = victim.getHealth() -
victim.getTiredness() + victim.getAge();
        maleFight = this.getHealth() - this.getTiredness() +
this.getAge();

        /* 3 possible resoulutions, outright win by male or
victim or draw */

        if(maleFight > victimFight){ victim.Kill(); }
        else if(maleFight < victimFight){ this.Kill(); }

        this.Breed();
        /* breed notifies class that it has finished by
setting the hasBreed variable to true */

    }

}

}

/*
=====
Overwritten functions
=====
*/
/* changed descriptions to allow personality among diffrent
tamagotchis
added new states FIGHTING and BREEDING */
public String getStateDescpt(){

    String ret_string = "no state defined";

    if (_state == BORED){
        ret_string = getName() + " is bored. Do something.";
    }
    else if(_state == HUNGRY){
        ret_string = getName() + " needs his food...";
    }
    else if(_state == TIRED){

```

```

        ret_string = getName() + " wants to goto bed now.";
    }
    else if(_state == ASLEEP){
        ret_string = getName() + ", he is snoring in his sleep.";
    }
    else if(_state == DEAD){
        ret_string = getName() + " died. :{";
    }
    else if(_state == OK){
        ret_string = getName() + ", he is happy! :}";
    }
    /* 2 new states - fighting and breeding */
    else if(_state == FIGHTING){
        ret_string = getName() + " is angry and he wants to
fight!";
    }
    else if(_state == BREEDING){
        ret_string = getName() + " wishes to continue his line.";
    }
    else {
        ret_string = new String("Error: Invalid State.");
    }

    return ret_string;
}
/* updated this function so it can handle the 2 new states! */
protected void setState(int s){ if(s >= 0 && s <= 7) _state = s; }

}

```

TamaManager.java

```
////////////////////////////////////
/
//
// File: TamaManager.java
//
// Written by Mark Cunningham 95309021
// unless otherwise stated.
//
// Compiler:          javac; JDK 1.1.6 (and JDK 1.1.5)
// Operating System used:  Windows 95 and Windows NT
//
// From today 4th December 1998 everything complies
// and bugs as reported in code
//
////////////////////////////////////
/

import java.lang.*;          /* basic import class */
import TamaInterface;       /* frontend to TamaManager */
import TamaBasic;           /* basic Tamagotchi */
import TamaMale;            /* male Tamagotchi */
import TamaFemale;          /* female Tamagotchi */

////////////////////////////////////
/
//
// This is the Tamagotchi Manager.
// It is the virtual world of the Tamagotchis.
//
// It deals with inter Tamagotchi reactions, such as
// breeding and fighting. Removes dead tamagotchis and
// allows the adding of new tamagotchis.
//
// It deals with a maximum of only 4 tamagotchis, this
// is due to the current limitation of the TamaInterface frontend.
//
// The position in the TamaInterface represents the internal
// 'id' of the Tamagotchi here.
//
// It also acts as the go-between the interface and the
tamagotchis.
//
// It is implemented as a thread to allow faster processing and
// to leave it independant of wheater it's an applet or
// application. Though it might be ideal to have the tamagotchis
// individually as threads it can easily leads to messy protocols
and
// runaway threads.
//
////////////////////////////////////
/
```

```

public class TamaManager extends Object implements Runnable{

    /* maximum number of Tamagotchis */
    private static int MAX_TAMAGOTCHIS = 4;
    /* frontend */
    private TamaInterface frontend;
    /* array of pointers to the tamagotchis */
    private TamaBasic tamagotchis[] = new TamaBasic[MAX_TAMAGOTCHIS];
    /* the thread */
    public Thread tamay;

    /* constructor */
    public TamaManager(){

        /* init frontend */
        frontend = new TamaInterface(this);

        /* init tamagotchis to null */
        int cnt = 0;
        for(cnt = 0; cnt < MAX_TAMAGOTCHIS; cnt++){
            tamagotchis[cnt] = null;
        }
    }

    /*
    =====
    Internal Functions
    =====
    */

    /* updates and deals with tamagotchis */
    private void update(){

        /* go through each tamagotchi */
        int cnt = 0;
        for(cnt = 0; cnt < MAX_TAMAGOTCHIS; cnt++){

            if(tamagotchis[cnt] != null){

                /* if valid tamagotchi, update */
                tamagotchis[cnt].update();

                /* message */
                frontend.displayMessage("\n[" + cnt + "] " +
                tamagotchis[cnt].getStateDescpt());

                /* debug */ System.out.println("[DEBUG] (TamaManager) "
                + tamagotchis[cnt].getName() + " (" + tamagotchis[cnt].getType() +
                ") [" + cnt + "]: health; " + tamagotchis[cnt].getHealth() + "
                Tiredness; " + tamagotchis[cnt].getTiredness() + " Bored; " +
                tamagotchis[cnt].getBored() + " Age; " +
                tamagotchis[cnt].getAge());

                /* check male actions: fighting and breeding */

```

```

        checkMale(cnt);

        /* if dead tamagotchi remove and clean up */
        if (tamagotchis[cnt].getState() == TamaBasic.DEAD){
            tamagotchis[cnt] = null;
            frontend.removeTama(cnt);
        }
    }
}

/* checks if male and completes breeding and action states of
tamagotchi */
private void checkMale(int ID){

    /* if male it can breed and fight */
    if (tamagotchis[ID].getType().equals("MALE")){

        /* check if he wants to breeding */
        if(tamagotchis[ID].getState() == TamaMale.BREEDING){

            int cnt1 = 0;
            for(; cnt1 < MAX_TAMAGOTCHIS; cnt1++){

                if(tamagotchis[cnt1] != null &&
tamagotchis[cnt1].getType().equals("FEMALE")){

                    /* male and female breed! */
                    ((TamaMale)tamagotchis[ID]).Breed();
                    newTama(((TamaFemale)tamagotchis[cnt1]).Breed());

                    frontend.displayMessage("\n" +
tamagotchis[ID].getName() + " [" + ID + "] is attempting to breed
with " + tamagotchis[cnt1].getName() + "[" + cnt1 + "]");

                    cnt1 = MAX_TAMAGOTCHIS; /* escapes loop */

                }
            }
        }
        /* check if he's fighting */
        else if(tamagotchis[ID].getState() == TamaMale.FIGHTING){

            int cnt1 = MAX_TAMAGOTCHIS - 1;

            for(; cnt1 >= 0; cnt1--){

                if(tamagotchis[cnt1] != null && tamagotchis[cnt1]
!= tamagotchis[ID]){

                    /* pick a fight */

                    ((TamaMale)tamagotchis[ID]).Fight((TamaBasic)tamagotchis[cnt1]);

```

```

        frontend.displayMessage("\n" +
tamagotchis[ID].getName() + " [" + ID + "] is fighting with " +
tamagotchis[cnt1].getName() + "[" + cnt1 + "]");

        cnt1 = -1; /* to escape loop */
    }

    } /* end for */

    } /* end else if */

    } /* end main if */

} /* end function */

/*
=====
Implements the runnable interface
=====
*/

public void start(){

    /* make frontend visible and start thread */
    frontend.setVisible(true);
    frontend.displayMessage("\n** Welcome to Tamagotchi Families
**\n");
    tamay = new Thread(this);
    if (tamay != null) { tamay.start(); }

}

public void run() {

    while(true){
        update(); /* continously update tamagotchis */
        Thread.currentThread().yield();
        try { Thread.sleep(2000); }
        catch (Exception exc) {};
    }

}

public void stop() {

    if (tamay != null) {
        tamay.stop();
        tamay = null;
    }

}

/*
=====
Public functions available to Frontend

```

```

=====
*/
/* play with specific tamagotchi */
public void Play(int ID){
    if(tamagotchis[ID] != null){
        tamagotchis[ID].Play();
    }
}
/* feed specific tamagotchi */
public void Feed(int ID){
    if(tamagotchis[ID] != null){
        tamagotchis[ID].Feed();
    }
}
/* sleep specific tamagotchi */
public void Sleep(int ID){
    if(tamagotchis[ID] != null){
        tamagotchis[ID].Sleep();
    }
}
/* kill specific tamagotchi */
public void Kill(int ID){
    if(tamagotchis[ID] != null){
        tamagotchis[ID].Kill();
    }
}
/* test specific tamagotchi - change it's internal
variables to make it enter some state */
public void test(int ID){
    if(tamagotchis[ID] != null){
        tamagotchis[ID].setHealth(20);
        tamagotchis[ID].setBored(0);
    }
}
/* add a new tamagotchi */
public void newTama(){

    TamaBasic newBaby = null;
    double randomType = Math.random();

    /* create a random tamagotchi */
    if(randomType < 0.3){ newBaby = new TamaBasic(20, 0, 0, 3,
"Norm"); }
    else if(randomType < 0.6){ newBaby = new TamaFemale(20, 0, 0,
3, "Girl"); }
    else if(randomType <= 1){ newBaby = new TamaMale(20, 0, 0, 3,
"Boy"); }

    /* add it */
    newTama(newBaby);
}
public void newTama(TamaBasic newBaby){

    if(newBaby != null){

```

```
int ID = -1;
ID = frontend.addTama(newBaby.getName());

if(ID != -1){
    /* if front end accepts it, add tamagotchi to Manager */
    tamagotchis[ID] = newBaby;
    frontend.displayMessage("\nNew tamagotchi created: " +
newBaby.getName() + " [" + ID + "]");
}
}
}
```

TamaApplication.java

```
////////////////////////////////////
/
//
// File: TamaApplication.java
//
// Written by Mark Cunningham 95309021
// unless otherwise stated.
//
// Compiler:          javac; JDK 1.1.6 (and JDK 1.1.5)
// Operating System used:  Windows 95 and Windows NT
//
// From today 4th December 1998 everything complies
// and bugs as reported in code
//
////////////////////////////////////
/

import java.lang.*; /* basic java import */
import TamaManager; /* Tamagotchi controller */

////////////////////////////////////
/
//
// the main purpose of this class it to
// start the tamgotchi manager thread
//
////////////////////////////////////
/

class TamaApplication {

public static void main(String[] args) {

    /* new TamaManager */
    TamaManager tamagotchis = new TamaManager();

    /* start TamaManager */
    tamagotchis.start();

    /* add a Tamagotchi to TamaManager to get the
    ball rolling */
    tamagotchis.newTama();

}

}
```

TamaInterface.java

```
////////////////////////////////////
/
//
// File: TamaInterface.java
//
// Written by Mark Cunningham 95309021
// unless otherwise stated.
//
// Compiler:          javac; JDK 1.1.6 (and JDK 1.1.5)
// Operating System used:  Windows 95 and Windows NT
//
// From today 4th December 1998 everything complies
// and bugs as reported in code
//
////////////////////////////////////
/

import java.lang.*;      /* basic import class */
import java.awt.*;      /* for the api */
import java.awt.event.*; /* for event interface: ActionListener */
import TamaManager;     /* TamaManager class */

////////////////////////////////////
/
//
// This class is the user's interface to the Tamagotchi Manager.
//
// It seperates the JAVA AWT code away from the Tamagotchi
// implementation code.
//
// It and the TamaManager suffer a design flaw in that
// they both mirror certain aspects internal, ie the number
// of tamagotchis at one time is depentant on what TamaInterface
// can show. With more time a better frontend could be
// designed to allow unlimited Tamagotchis running.
//
// It also doesn't allow full customisation of tamagotchis.
// While the functions in the tamagotchis exist, a user
// interface to them hasn't been created. Example, there could
// be a menu bar to specify maximum health, a button to allow
// the setting of names etc.
//
////////////////////////////////////
/

/*
This class is a frame that acts like an Application Window,
it involves all code to deal with buttons and such.
*/
public class TamaInterface extends Frame implements
ActionListener{
```

```

/* check boxes and names to display the tamagotchis available
   and to be able to select specific ones */
private Checkbox TamaCBox [] = new Checkbox[4];
private TextField TamaText [] = new TextField[4];

/* main text area for displaying output to user */
private TextArea MainTextA;

/* the Tamagotchi Manager */
private TamaManager Manager;

/* constructor */
public TamaInterface(TamaManager M){

    /* super class constructor */
    super("Tamagotchi Families");

    /* set size of window */
    setSize(600,300);

    /* set the "layout" manager to FlowLayout */
    setLayout(new FlowLayout());

    /* this panel holds main administration buttons:
       new tamagotchi and quit */
    Panel panell = new Panel();
    add(panell);

    /* new tamagotchi button */
    Button newBut = new Button("New");
    newBut.addActionListener(this);
    panell.add(newBut);

    /* quit button :- forces a system.exit */
    Button quitBut = new Button("Quit");
    quitBut.addActionListener(this);
    panell.add(quitBut);

    /* this panel holds all the user buttons
       play, feed, kill, sleep */
    Panel panel2 = new Panel();
    add(panel2);

    /* special test button, left it in as
       a feature. It allows testing of tamagotchi breeding
       by uping the stats of the specific tamagotchi */
    Button testState = new Button("Drug");
    testState.addActionListener(this);
    panel2.add(testState);

    /* play button */
    Button playBut = new Button("Play");
    playBut.addActionListener(this);
    panel2.add(playBut);

```

```

/* feed button */
Button feedBut = new Button("Feed");
feedBut.addActionListener(this);
panel2.add(feedBut);

/* sleep button */
Button sleepBut = new Button("Sleep");
sleepBut.addActionListener(this);
panel2.add(sleepBut);

/* kill button */
Button killBut = new Button("Kill");
killBut.addActionListener(this);
panel2.add(killBut);

/* this panel holds information on
   the tamagotchis and a way to select which one
   to use */
Panel panel3 = new Panel();
add(panel3);

/* there is textfield holding the tamagotchi's name
   beside a checkbox, click the check box to select
   that tamagotchi. The checkboxgroup means that
   only one tamagotchi is selected. If a tamgotchi
   isn't there, the textfield and checkbox are
   disabled.

   Each tamagotch is given an ID dependant on what
   checkbox they get - there is only four tamagotchis
   running at one time
*/
CheckboxGroup whichTamaCBoxG = new CheckboxGroup();
int cnt = 0;
for(; cnt < 4; cnt++){
    TamaCBox[cnt] = new Checkbox(" " + cnt + " ",
whichTamaCBoxG, false);
    TamaCBox[cnt].setEnabled(false);
    TamaText[cnt] = new TextField(10);
    TamaText[cnt].setEnabled(false);
    TamaText[cnt].setEditable(false);
    panel3.add(TamaText[cnt]); panel3.add(TamaCBox[cnt]);
}

/* this panel holds the output textbox to user showing the
current
   states of the running tamagotchis */
Panel panel4 = new Panel();
add(panel4);

/* text area */
MainTextA = new TextArea("", 10, 50,
TextArea.SCROLLBARS_VERTICAL_ONLY);
MainTextA.setEditable(false);
panel4.add(MainTextA);

```

```

    /* pointer to manager class */
    Manager = M;

}

/*
=====
implements ActionListener interface
=====
*/
public void actionPerformed(ActionEvent evt){

    String eventString = evt.getActionCommand();

    /* finish program */
    if(eventString.equals("Quit")){
        System.out.println("[DEBUG] (TamaInterface) Quit :- exiting
program...");
        System.exit(0);
    }
    /* add a new tamagotchi */
    else if(eventString.equals("New")){
        System.out.println("[DEBUG] (TamaInterface) New :-
attempting to add new Tamagothic...");
        Manager.newTama(); /* call manager class function to create a
tamagotchi */
    }
    else {

        /* if not administration function, then a function on the
tamagotchi's themselves */

        /* find which tamagotchi is selected */
        int id = 0;
        for(; id < 4 && !TamaCBox[id].getState(); id++){

            /* as long as valid - if no tamagotchi's running then this if
statement fails */
            if(id < 4){

                if(eventString.equals("Play")){
                    Manager.Play(id);
                    System.out.println("[DEBUG] (TamaInterface) Play :-
play method on Tamagothic [" + id + "]");
                }
                else if(eventString.equals("Kill")){
                    Manager.Kill(id);
                    System.out.println("[DEBUG] (TamaInterface) Kill :-
kill method on Tamagothic [" + id + "]");
                }
                else if(eventString.equals("Feed")){
                    Manager.Feed(id);
                    System.out.println("[DEBUG] (TamaInterface) Feed :-
feed method on Tamagothic [" + id + "]");
                }
            }
        }
    }
}

```

```

        }
        else if(eventString.equals("Sleep")){
            System.out.println("[DEBUG] (TamaInterface) Sleep :-
Sleep method on Tamagothic [" + id + "]);
            Manager.Sleep(id);
        }
        else if(eventString.equals("Drug")){
            System.out.println("[DEBUG] (TamaInterface) Drug :-
test state on Tamagothic [" + id + "]);
            Manager.test(id);
        }
    }
}

}

}

/*
=====
public access functions to the TamaManager class
=====
*/

/* display a message to the text output */
public void displayMessage(String Message){
    MainTextA.append(Message);
    /* debug */ System.out.println("[DEBUG] (TamaInterface) Message
Displayed: " + Message);
}

/* remove a tamagotchi from the frontend */
public boolean removeTama(int ID){

    boolean returnValue = false;

    if(ID >= 0 && ID < 4){
        TamaText[ID].setText("");
        TamaText[ID].setEnabled(false);
        TamaCBox[ID].setState(false);
        TamaCBox[ID].setEnabled(false);
        returnValue = true;
        System.out.println("[DEBUG] (TamaInterface) Tamagothic
Removed [" + ID + "]);
    }

    return returnValue;
}

/* add a tamagotchi: sees if theres an empty slot, if so
adds it and returns it's ID, if not returns -1 and
does nothing */
public int addTama(String newName){

    int cnt = 0, returnID = -1;
    for(; cnt < 4 && TamaCBox[cnt].isEnabled(); cnt++){

```

```

    if(cnt < 4){
        TamaText[cnt].setText(newName);
        TamaText[cnt].setEnabled(true);
        TamaCBox[cnt].setState(true);
        TamaCBox[cnt].setEnabled(true);
        returnID = cnt;
        System.out.println("[DEBUG] (TamaInterface) New Tamagothic
Added [" + cnt + "]: " + newName);
    }

    return returnID;

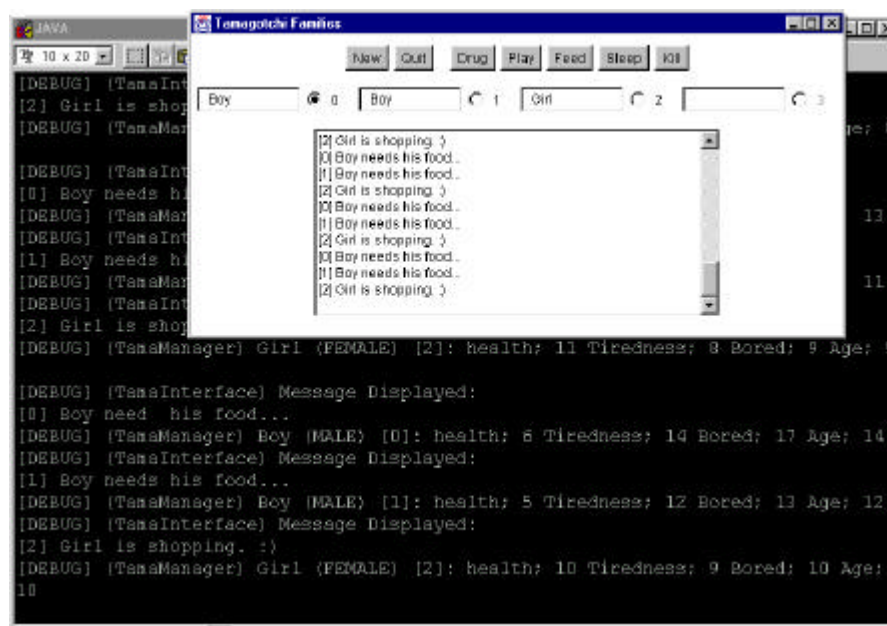
}

}

```

Sample output/input session

View of Frontend while program is running.



Sample DEBUG text Output from a simple session:-

```
[DEBUG] (TamaInterface) Message Displayed:
** Welcome to Tamagotchi Families **

[DEBUG] (TamaInterface) New Tamagothic Added [0]: Norm
[DEBUG] (TamaInterface) Message Displayed:
New tamagotchi created: Norm [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 19 Tiredness; 1 Bored; 1 Age; 1
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 18 Tiredness; 2 Bored; 2 Age; 2
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 17 Tiredness; 3 Bored; 3 Age; 3
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 16 Tiredness; 4 Bored; 4 Age; 4
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 15 Tiredness; 5 Bored; 5 Age; 5
[DEBUG] (TamaInterface) Play :- play method on Tamagothic [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 14 Tiredness; 6 Bored; 9 Age; 6
[DEBUG] (TamaInterface) Play :- play method on Tamagothic [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 13 Tiredness; 7 Bored; 13 Age; 7
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 12 Tiredness; 8 Bored; 14 Age; 8
[DEBUG] (TamaInterface) Feed :- feed method on Tamagothic [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 11 Tiredness; 9 Bored; 15 Age; 9
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Bored...
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 10 Tiredness; 10 Bored; 16 Age; 10
[DEBUG] (TamaInterface) Play :- play method on Tamagothic [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 9 Tiredness; 11 Bored; 14 Age; 11
[DEBUG] (TamaInterface) Feed :- feed method on Tamagothic [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 11 Tiredness; 12 Bored; 15 Age; 12
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Bored...
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 10 Tiredness; 13 Bored; 16 Age; 13
[DEBUG] (TamaInterface) Play :- play method on Tamagothic [0]
```

```

[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 9 Tiredness; 14 Bored; 14 Age; 14
[DEBUG] (TamaInterface) Feed :- feed method on Tamagothic [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 11 Tiredness; 15 Bored; 15 Age; 15
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Sleepy...
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 10 Tiredness; 16 Bored; 16 Age; 16
[DEBUG] (TamaInterface) Sleep :- Sleep method on Tamagothic [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Sleeping...
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 11 Tiredness; 15 Bored; 15 Age; 17
[DEBUG] (TamaInterface) New :- attempting to add new Tamagothic...
[DEBUG] (TamaInterface) New Tamagothic Added [1]: Norm
[DEBUG] (TamaInterface) Message Displayed:
New tamagotchi created: Norm [1]
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Sleeping...
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 12 Tiredness; 14 Bored; 14 Age; 18
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 19 Tiredness; 1 Bored; 1 Age; 1
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Sleeping...
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 13 Tiredness; 13 Bored; 13 Age; 19
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 18 Tiredness; 2 Bored; 2 Age; 2
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Sleeping...
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; 14 Tiredness; 12 Bored; 12 Age; 20
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 17 Tiredness; 3 Bored; 3 Age; 3
[DEBUG] (TamaInterface) Kill :- kill method on Tamagothic [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Norm is Dead.
[DEBUG] (TamaManager) Norm (BASIC) [0]: health; -10 Tiredness; 12 Bored; 12 Age; 20
[DEBUG] (TamaInterface) Tamagothic Removed [0]
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 16 Tiredness; 4 Bored; 4 Age; 4
[DEBUG] (TamaInterface) New :- attempting to add new Tamagothic...
[DEBUG] (TamaInterface) New Tamagothic Added [0]: Boy
[DEBUG] (TamaInterface) Message Displayed:
New tamagotchi created: Boy [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy, he is happy! :}
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 19 Tiredness; 1 Bored; 1 Age; 1
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Okay!
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 15 Tiredness; 5 Bored; 5 Age; 5
[DEBUG] (TamaInterface) Message Displayed:

```

[0] Boy, he is happy! : }

[DEBUG] (TamaManager) Boy (MALE) [0]: health; 18 Tiredness; 2 Bored; 2 Age; 2

[DEBUG] (TamaInterface) Message Displayed:

[1] Norm is Okay!

[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 14 Tiredness; 6 Bored; 6 Age; 6

[DEBUG] (TamaInterface) New :- attempting to add new Tamagothic...

[DEBUG] (TamaInterface) New Tamagothic Added [2]: Norm

[DEBUG] (TamaInterface) Message Displayed:

New tamagotchi created: Norm [2]

[DEBUG] (TamaInterface) Message Displayed:

[0] Boy, he is happy! : }

[DEBUG] (TamaManager) Boy (MALE) [0]: health; 17 Tiredness; 3 Bored; 3 Age; 3

[DEBUG] (TamaInterface) Message Displayed:

[1] Norm is Okay!

[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 13 Tiredness; 7 Bored; 7 Age; 7

[DEBUG] (TamaInterface) Message Displayed:

[2] Norm is Okay!

[DEBUG] (TamaManager) Norm (BASIC) [2]: health; 19 Tiredness; 1 Bored; 1 Age; 1

[DEBUG] (TamaInterface) Kill :- kill method on Tamagothic [2]

[DEBUG] (TamaInterface) Message Displayed:

[0] Boy, he is happy! : }

[DEBUG] (TamaManager) Boy (MALE) [0]: health; 16 Tiredness; 4 Bored; 4 Age; 4

[DEBUG] (TamaInterface) Message Displayed:

[1] Norm is Okay!

[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 12 Tiredness; 8 Bored; 8 Age; 8

[DEBUG] (TamaInterface) Message Displayed:

[2] Norm is Dead.

[DEBUG] (TamaManager) Norm (BASIC) [2]: health; -10 Tiredness; 1 Bored; 1 Age; 1

[DEBUG] (TamaInterface) Tamagothic Removed [2]

[DEBUG] (TamaInterface) New :- attempting to add new Tamagothic...

[DEBUG] (TamaInterface) New Tamagothic Added [2]: Girl

[DEBUG] (TamaInterface) Message Displayed:

New tamagotchi created: Girl [2]

[DEBUG] (TamaInterface) Message Displayed:

[0] Boy, he is happy! : }

[DEBUG] (TamaManager) Boy (MALE) [0]: health; 15 Tiredness; 5 Bored; 5 Age; 5

[DEBUG] (TamaInterface) Message Displayed:

[1] Norm is Okay!

[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 11 Tiredness; 9 Bored; 9 Age; 9

[DEBUG] (TamaInterface) Message Displayed:

[2] Girl is shopping. :)

[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 19 Tiredness; 1 Bored; 1 Age; 1

[DEBUG] (TamaInterface) Message Displayed:

[0] Boy, he is happy! : }

[DEBUG] (TamaManager) Boy (MALE) [0]: health; 14 Tiredness; 6 Bored; 6 Age; 6

[DEBUG] (TamaInterface) Message Displayed:

[1] Norm is Okay!

[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 10 Tiredness; 10 Bored; 10 Age; 10

[DEBUG] (TamaInterface) Message Displayed:

[2] Girl is shopping. :)

[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 18 Tiredness; 2 Bored; 2 Age; 2

[DEBUG] (TamaInterface) Message Displayed:

[0] Boy, he is happy! : }

[DEBUG] (TamaManager) Boy (MALE) [0]: health; 13 Tiredness; 7 Bored; 7 Age; 7

[DEBUG] (TamaInterface) Message Displayed:

[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 9 Tiredness; 11 Bored; 11 Age; 11
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl is shopping. :)
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 17 Tiredness; 3 Bored; 3 Age; 3
[DEBUG] (TamaInterface) Play :- play method on Tamagothic [1]
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy, he is happy! :}
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 12 Tiredness; 8 Bored; 8 Age; 8
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 8 Tiredness; 12 Bored; 12 Age; 12
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl is shopping. :)
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 16 Tiredness; 4 Bored; 4 Age; 4
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy, he is happy! :}
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 11 Tiredness; 9 Bored; 9 Age; 9
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 7 Tiredness; 13 Bored; 13 Age; 13
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl is shopping. :)
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 15 Tiredness; 5 Bored; 5 Age; 5
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy, he is happy! :}
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 10 Tiredness; 10 Bored; 10 Age; 10
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 6 Tiredness; 14 Bored; 14 Age; 14
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl is shopping. :)
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 14 Tiredness; 6 Bored; 6 Age; 6
[DEBUG] (TamaInterface) Feed :- feed method on Tamagothic [1]
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy needs his food...
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 9 Tiredness; 11 Bored; 11 Age; 11
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 8 Tiredness; 15 Bored; 15 Age; 15
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl is shopping. :)
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 13 Tiredness; 7 Bored; 7 Age; 7
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy needs his food...
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 8 Tiredness; 12 Bored; 12 Age; 12
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 7 Tiredness; 16 Bored; 16 Age; 16
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl is shopping. :)
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 12 Tiredness; 8 Bored; 8 Age; 8
[DEBUG] (TamaInterface) Feed :- feed method on Tamagothic [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy needs his food...

[DEBUG] (TamaManager) Boy (MALE) [0]: health; 10 Tiredness; 13 Bored; 13 Age; 13
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 6 Tiredness; 17 Bored; 17 Age; 17
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl is shopping. :)
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 11 Tiredness; 9 Bored; 9 Age; 9
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy needs his food...
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 9 Tiredness; 14 Bored; 14 Age; 14
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 5 Tiredness; 18 Bored; 18 Age; 18
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl is shopping. :)
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 10 Tiredness; 10 Bored; 10 Age; 10
[DEBUG] (TamaInterface) Feed :- feed method on Tamagothic [1]
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy needs his food...
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 8 Tiredness; 15 Bored; 15 Age; 15
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 7 Tiredness; 19 Bored; 19 Age; 19
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl wants her chocolate and ice-cream.
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 9 Tiredness; 11 Bored; 11 Age; 11
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy needs his food...
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 7 Tiredness; 16 Bored; 16 Age; 16
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 6 Tiredness; 20 Bored; 20 Age; 20
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl wants her chocolate and ice-cream.
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 8 Tiredness; 12 Bored; 12 Age; 12
[DEBUG] (TamaInterface) Feed :- feed method on Tamagothic [2]
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy needs his food...
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 6 Tiredness; 17 Bored; 17 Age; 17
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 5 Tiredness; 21 Bored; 21 Age; 21
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl is shopping. :)
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 10 Tiredness; 13 Bored; 13 Age; 13
[DEBUG] (TamaInterface) Feed :- feed method on Tamagothic [1]
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy needs his food...
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 5 Tiredness; 18 Bored; 18 Age; 18
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 7 Tiredness; 22 Bored; 22 Age; 22
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl wants her chocolate and ice-cream.
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 9 Tiredness; 14 Bored; 14 Age; 14

[DEBUG] (TamaInterface) Play :- play method on Tamagothic [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy needs his food...
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 4 Tiredness; 19 Bored; 16 Age; 19
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 6 Tiredness; 23 Bored; 23 Age; 23
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl wants her chocolate and ice-cream.
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 8 Tiredness; 15 Bored; 15 Age; 15
[DEBUG] (TamaInterface) Drug :- test state on Tamagothic [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy wishes to continue his line.
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 19 Tiredness; 20 Bored; 1 Age; 20
[DEBUG] (TamaInterface) New Tamagothic Added [3]: Baby Boy
[DEBUG] (TamaInterface) Message Displayed:
New tamagotchi created: Baby Boy [3]
[DEBUG] (TamaInterface) Message Displayed:
Boy [0] is attempting to breed with Girl[2]
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 5 Tiredness; 24 Bored; 24 Age; 24
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl wants her chocolate and ice-cream.
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 3 Tiredness; 16 Bored; 16 Age; 16
[DEBUG] (TamaInterface) Message Displayed:
[3] Baby Boy, he is happy! :}
[DEBUG] (TamaManager) Baby Boy (MALE) [3]: health; 19 Tiredness; 1 Bored; 1 Age; 1
[DEBUG] (TamaInterface) Feed :- feed method on Tamagothic [2]
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy wants to goto bed now.
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 18 Tiredness; 21 Bored; 2 Age; 21
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 4 Tiredness; 25 Bored; 25 Age; 25
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl wants her chocolate and ice-cream.
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 5 Tiredness; 17 Bored; 17 Age; 17
[DEBUG] (TamaInterface) Message Displayed:
[3] Baby Boy, he is happy! :}
[DEBUG] (TamaManager) Baby Boy (MALE) [3]: health; 18 Tiredness; 2 Bored; 2 Age; 2
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy wishes to continue his line.
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 17 Tiredness; 22 Bored; 3 Age; 22
[DEBUG] (TamaInterface) Message Displayed:
Boy [0] is attempting to breed with Girl[2]
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 3 Tiredness; 26 Bored; 26 Age; 26
[DEBUG] (TamaInterface) Message Displayed:
[2] Girl died. :(
[DEBUG] (TamaManager) Girl (FEMALE) [2]: health; 5 Tiredness; 17 Bored; 17 Age; 18
[DEBUG] (TamaInterface) Tamagothic Removed [2]
[DEBUG] (TamaInterface) Message Displayed:
[3] Baby Boy, he is happy! :}

[DEBUG] (TamaManager) Baby Boy (MALE) [3]: health; 17 Tiredness; 3 Bored; 3 Age; 3
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy wants to goto bed now.
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 16 Tiredness; 23 Bored; 4 Age; 23
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 2 Tiredness; 27 Bored; 27 Age; 27
[DEBUG] (TamaInterface) Message Displayed:
[3] Baby Boy, he is happy! :}
[DEBUG] (TamaManager) Baby Boy (MALE) [3]: health; 16 Tiredness; 4 Bored; 4 Age; 4
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy wants to goto bed now.
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 15 Tiredness; 24 Bored; 5 Age; 24
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 1 Tiredness; 28 Bored; 28 Age; 28
[DEBUG] (TamaInterface) Message Displayed:
[3] Baby Boy, he is happy! :}
[DEBUG] (TamaManager) Baby Boy (MALE) [3]: health; 15 Tiredness; 5 Bored; 5 Age; 5
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy wants to goto bed now.
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 14 Tiredness; 25 Bored; 6 Age; 25
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Hungry...
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 0 Tiredness; 29 Bored; 29 Age; 29
[DEBUG] (TamaInterface) Message Displayed:
[3] Baby Boy, he is happy! :}
[DEBUG] (TamaManager) Baby Boy (MALE) [3]: health; 14 Tiredness; 6 Bored; 6 Age; 6
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy wants to goto bed now.
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 13 Tiredness; 26 Bored; 7 Age; 26
[DEBUG] (TamaInterface) Message Displayed:
[1] Norm is Dead.
[DEBUG] (TamaManager) Norm (BASIC) [1]: health; 0 Tiredness; 29 Bored; 29 Age; 29
[DEBUG] (TamaInterface) Tamagothic Removed [1]
[DEBUG] (TamaInterface) Message Displayed:
[3] Baby Boy, he is happy! :}
[DEBUG] (TamaManager) Baby Boy (MALE) [3]: health; 13 Tiredness; 7 Bored; 7 Age; 7
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy wants to goto bed now.
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 12 Tiredness; 27 Bored; 8 Age; 27
[DEBUG] (TamaInterface) Message Displayed:
[3] Baby Boy, he is happy! :}
[DEBUG] (TamaManager) Baby Boy (MALE) [3]: health; 12 Tiredness; 8 Bored; 8 Age; 8
[DEBUG] (TamaInterface) Kill :- kill method on Tamagothic [3]
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy wants to goto bed now.
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 11 Tiredness; 28 Bored; 9 Age; 28
[DEBUG] (TamaInterface) Message Displayed:
[3] Baby Boy died. :{
[DEBUG] (TamaManager) Baby Boy (MALE) [3]: health; -10 Tiredness; 8 Bored; 8 Age; 8
[DEBUG] (TamaInterface) Tamagothic Removed [3]
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy wants to goto bed now.
[DEBUG] (TamaManager) Boy (MALE) [0]: health; 10 Tiredness; 29 Bored; 10 Age; 29

```
[DEBUG] (TamaInterface) Kill :- kill method on Tamagothic [0]
[DEBUG] (TamaInterface) Message Displayed:
[0] Boy died. :{
[DEBUG] (TamaManager) Boy (MALE) [0]: health; -10 Tiredness; 29 Bored; 10 Age; 29
[DEBUG] (TamaInterface) Tamagothic Removed [0]
[DEBUG] (TamaInterface) Quit :- exiting program...
```

Sample User text Output from the same session:-

** Welcome to Tamagotchi Families **

New tamagotchi created: Norm [0]

```
[0] Norm is Okay!
[0] Norm is Okay!
[0] Norm is Okay!
[0] Norm is Okay!
[0] Norm is Okay!
[0] Norm is Okay!
[0] Norm is Okay!
[0] Norm is Okay!
[0] Norm is Okay!
[0] Norm is Bored...
[0] Norm is Hungry...
[0] Norm is Okay!
[0] Norm is Bored...
[0] Norm is Hungry...
[0] Norm is Okay!
[0] Norm is Sleepy...
[0] Norm is Sleeping...
```

New tamagotchi created: Norm [1]

```
[0] Norm is Sleeping...
[1] Norm is Okay!
[0] Norm is Sleeping...
[1] Norm is Okay!
[0] Norm is Sleeping...
[1] Norm is Okay!
[0] Norm is Dead.
[1] Norm is Okay!
```

New tamagotchi created: Boy [0]

```
[0] Boy, he is happy! :}
[1] Norm is Okay!
[0] Boy, he is happy! :}
[1] Norm is Okay!
```

New tamagotchi created: Norm [2]

```
[0] Boy, he is happy! :}
[1] Norm is Okay!
[2] Norm is Okay!
[0] Boy, he is happy! :}
[1] Norm is Okay!
[2] Norm is Dead.
```

New tamagotchi created: Girl [2]

```
[0] Boy, he is happy! :}
[1] Norm is Okay!
```

[2] Girl is shopping. :)
 [0] Boy, he is happy! :}
 [1] Norm is Okay!
 [2] Girl is shopping. :)
 [0] Boy, he is happy! :}
 [1] Norm is Hungry...
 [2] Girl is shopping. :)
 [0] Boy, he is happy! :}
 [1] Norm is Hungry...
 [2] Girl is shopping. :)
 [0] Boy, he is happy! :}
 [1] Norm is Hungry...
 [2] Girl is shopping. :)
 [0] Boy, he is happy! :}
 [1] Norm is Hungry...
 [2] Girl is shopping. :)
 [0] Boy needs his food...
 [1] Norm is Hungry...
 [2] Girl is shopping. :)
 [0] Boy needs his food...
 [1] Norm is Hungry...
 [2] Girl is shopping. :)
 [0] Boy needs his food...
 [1] Norm is Hungry...
 [2] Girl is shopping. :)
 [0] Boy needs his food...
 [1] Norm is Hungry...
 [2] Girl is shopping. :)
 [0] Boy needs his food...
 [1] Norm is Hungry...
 [2] Girl is shopping. :)
 [0] Boy needs his food...
 [1] Norm is Hungry...
 [2] Girl wants her chocolate and ice-cream.
 [0] Boy needs his food...
 [1] Norm is Hungry...
 [2] Girl wants her chocolate and ice-cream.
 [0] Boy needs his food...
 [1] Norm is Hungry...
 [2] Girl is shopping. :)
 [0] Boy needs his food...
 [1] Norm is Hungry...
 [2] Girl wants her chocolate and ice-cream.
 [0] Boy needs his food...
 [1] Norm is Hungry...
 [2] Girl wants her chocolate and ice-cream.
 [0] Boy wishes to continue his line.
 New tamagotchi created: Baby Boy [3]
 Boy [0] is attempting to breed with Girl[2]
 [1] Norm is Hungry...
 [2] Girl wants her chocolate and ice-cream.
 [3] Baby Boy, he is happy! :}
 [0] Boy wants to goto bed now.
 [1] Norm is Hungry...
 [2] Girl wants her chocolate and ice-cream.
 [3] Baby Boy, he is happy! :}
 [0] Boy wishes to continue his line.
 Boy [0] is attempting to breed with Girl[2]

[2] Girl died. :(
[3] Baby Boy, he is happy! :}
[0] Boy wants to goto bed now.
[1] Norm is Hungry...
[3] Baby Boy, he is happy! :}
[0] Boy wants to goto bed now.
[1] Norm is Hungry...
[3] Baby Boy, he is happy! :}
[0] Boy wants to goto bed now.
[1] Norm is Hungry...
[3] Baby Boy, he is happy! :}
[0] Boy wants to goto bed now.
[1] Norm is Dead.
[3] Baby Boy, he is happy! :}
[0] Boy wants to goto bed now.
[3] Baby Boy, he is happy! :}
[0] Boy wants to goto bed now.
[3] Baby Boy died. :{
[0] Boy wants to goto bed now.
[0] Boy died. :{

*Project completed Friday 4th December.
Mark Cunningham 95309021*