Assignment 3

OOP

Maxwell Maia

Student ID: 21236277

24/10/2022

Assignment Description

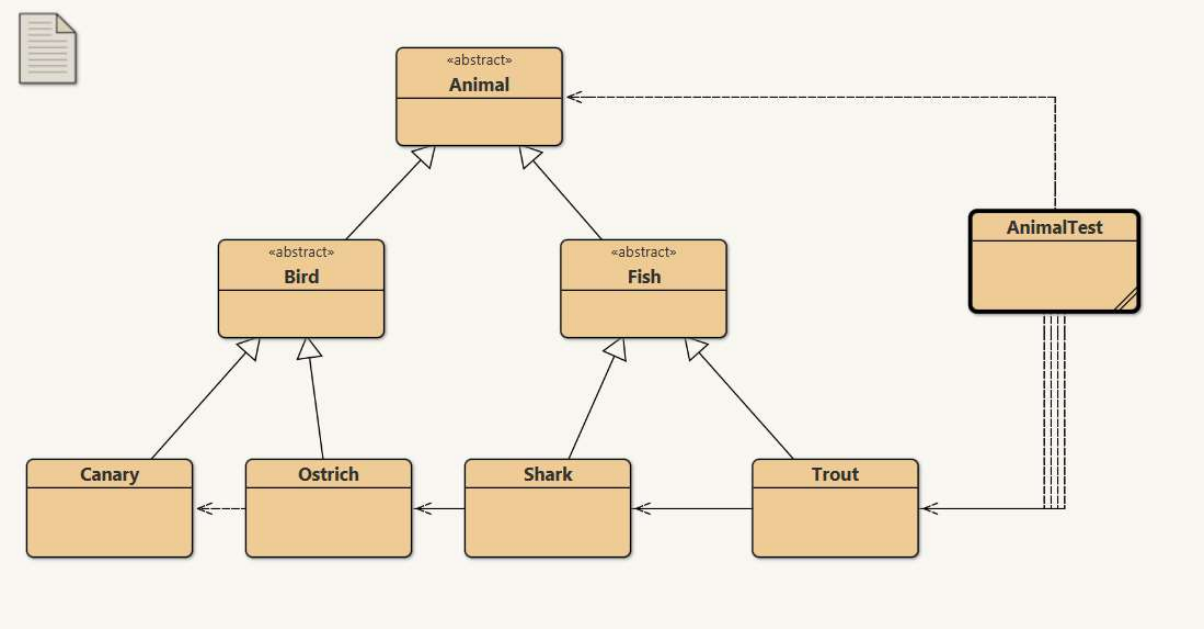
The point of this exercise is to practise inheritance.

Inheritance is used to avoid the repetition of attributes and methods.

Inheritance also allows polymorphism. Polymorphism is a property of an object that comes by an object being a one class of MANY classes that are inherited by a parent class. A reference variable of the parent or “super class” can store any of these objects.

Assignment description:

1. Make this inheritance tree.



2. Make custom toString() methods. This will print out information about the object.

3. Make custom equals(Object obj) methods. This will check whether two objects are the same or not by my own definition.

4. Make a move(int distance) method. For an Ostrich object the code should output “I am a bird but cannot fly” in the simplest way possible.

AnimalTest code

```
/**
 * @author Maxwell Maia
 * Class used to test the program.
 */
public class AnimalTest
{
    public AnimalTest()
    {

    }

    public static void main(String[] args)
    {
        //Create an object of the test class to run tests on.
        AnimalTest test = new AnimalTest();

        //Tests
        test.testToString(); //test1: demonstrate toString functionality
        test.testEquals(); //test2: demonstrate equals functionality
        test.testMove(); //demonstrate move and sing methods wor
    }

    //test1
    public void testToString()
    {
        System.out.println("=========Test 1=========");
        System.out.println("Test toString in all classes.\n\n");
```

```
        Animal[] animals = new Animal[4];
        //This declares and allocates memory for an array of Animal references with 4
spaces



        //Fill the Animals array with objects
        animals[0] = new Canary("Bluey");
        animals[1] = new Ostrich("Osty");
        animals[2] = new Shark("Hark");
        animals[3] = new Trout("Smekko");

        //For each element in the animal array
        // print out what is inside that element (toString)
        for(Animal animal : animals)
        {
            System.out.println(animal); //call toString() the animal
        }



    }

    //test2
    public void testEquals()
    {
        System.out.println("\n\n\n=========Test 2=========");
        System.out.println("Test the equals method in all classes.\n\n");



        //CANARY
        //Create an array for canaries
        Canary[] canary = new Canary[3];
```

```
        //Create some animals
        canary[0] = new Canary("jOhn");
        canary[1] = new Canary("jOhn");
        canary[2] = new Canary("Rizz");

        //.equals checking for Canary
        // equal
        System.out.println("Canary called " + canary[0].getName() +" vs " +
canary[1].getName() + " are they equal? ");
        System.out.println(canary[0].equals(canary[1]) ? "true" : "false");
        // not equal
        System.out.println("Canary called " + canary[0].getName() +" vs " +
canary[2].getName() + " are they equal? ");
        System.out.println(canary[0].equals(canary[2]) ? "true" : "false");

        //Formatting
        System.out.println("\n\n");



        //OSTRICH
        //Create an array for ostriches
        Ostrich[] ostrich = new Ostrich[3];

        //Create some animals
        ostrich[0] = new Ostrich("Maxwell");
        ostrich[1] = new Ostrich("Maxwell");
        ostrich[2] = new Ostrich("Ganyu");

        //.equals checking for Canary
```

```
        // equal
        System.out.println("Ostrich called " + ostrich[0].getName() +" vs " +
ostrich[1].getName() + " are they equal? ");
        System.out.println(ostrich[0].equals(ostrich[1]) ? "true" : "false");
        // not equal
        System.out.println("Ostrich called " + ostrich[0].getName() +" vs " +
ostrich[2].getName() + " are they equal? ");
        System.out.println(ostrich[0].equals(ostrich[2]) ? "true" : "false");

        //Formatting
        System.out.println("\n\n");



        //SHARK
        //Create an array for ostriches
        Shark[] shark = new Shark[3];

        //Create some animals
        shark[0] = new Shark("Yanfei");
        shark[1] = new Shark("Yanfei");
        shark[2] = new Shark("BOBBLES");

        //.equals checking for Canary
        // equal
        System.out.println("Shark called " + shark[0].getName() +" vs " +
shark[1].getName() + " are they equal? ");
        System.out.println(shark[0].equals(shark[1]) ? "true" : "false");
        // not equal
        System.out.println("Shark called " + shark[0].getName() +" vs " +
shark[2].getName() + " are they equal? ");
        System.out.println(shark[0].equals(shark[2]) ? "true" : "false");
```

```
        //Formatting
        System.out.println("\n\n");




        //TROUT
        //Create an array for ostriches
        Trout[] trout = new Trout[3];

        //Create some animals
        trout[0] = new Trout("Ahri");
        trout[1] = new Trout("Ahri");
        trout[2] = new Trout("Redox");

        //.equals checking for Canary
        // equal
        System.out.println("Trout called " + trout[0].getName() +" vs " +
trout[1].getName() + " are they equal? ");
        System.out.println(trout[0].equals(trout[1]) ? "true" : "false");
        // not equal
        System.out.println("Trout called " + trout[0].getName() +" vs " +
trout[2].getName() + " are they equal? ");
        System.out.println(trout[0].equals(trout[2]) ? "true" : "false");

        //Formatting
        System.out.println("\n\n");
```

```
        //Are different object types equal
        System.out.println("\nChecking that equals methods work for different types\n");

        Animal[] animals = new Animal[4];

        animals[0] = new Trout("Ahri");
        animals[1] = new Shark("Yanfei");
        animals[2] = new Canary("Rizz");
        animals[3] = new Ostrich("Maxwell");

        //.equals checking for different types
        // equal
        System.out.println("Trout called '" + animals[0].getName() +"' vs Trout called '" + animals[0].getName() + "' are they equal? ");
        System.out.println(animals[0].equals(animals[0]) ? "true" : "false");
        // not equal
        System.out.println("Trout called '" + animals[0].getName() +"' vs Shark called '" + animals[1].getName() + "' are they equal? ");
        System.out.println(animals[0].equals(animals[1]) ? "true" : "false");
        // not equal
        System.out.println("Ostrich called '" + animals[3].getName() +"' vs Trout called '" + animals[0].getName() + "' are they equal? ");
        System.out.println(animals[3].equals(animals[0]) ? "true" : "false");
        // not equal
        System.out.println("Ostrich called '" + animals[3].getName() +"' vs Canary called '" + animals[2].getName() + "' are they equal? ");
        System.out.println(animals[3].equals(animals[2]) ? "true" : "false");

    }
```

```
    public void testMove()
    {
        System.out.println("\n\n\n=========Test 3=========");
        System.out.println("Test move() method in all classes.\n\n");

        //Fill the Animals array with objects
        Animal[] animals = new Animal[4];
        animals[0] = new Canary("Bluey");
        animals[1] = new Ostrich("Osty");
        animals[2] = new Shark("Hark");
        animals[3] = new Trout("Smekko");

        //loop through animal array
        for(Animal animal : animals)
        {
            animal.move(5); //test move() method
            System.out.println(animal); //toString
        }
    }
}
```

AnimalTest test1 output

```
=========Test 1=========
Test toString in all classes.


Canary;
is a Bird; hasFeathers: true; hasWings: true; flies: true
is an Animal; name: Bluey; colour: yellow; hasSkin: true; breathes: true; eats: true;

Ostrich;
isTall: true; hasThinLongLegs: true;
is a Bird; hasFeathers: true; hasWings: true; flies: false
is an Animal; name: Osty; colour: black; hasSkin: true; breathes: true; eats: true;

Shark;
canBite: true; isDangerous: true;
is a Fish; hasFins: true; hasGills: true;
is an Animal; name: Hark; colour: grey; hasSkin: false; breathes: true; eats: true;

Trout;
hasSpikes: true; isEdible: true; swimsUpriverToLayEggs: true;
is a Fish; hasFins: true; hasGills: true;
is an Animal; name: Smekko; colour: brown; hasSkin: false; breathes: true; eats: true;
```

AnimalTest test2 output

```
=========Test 2=========
Test the equals method in all classes.


Canary called jOhn vs jOhn are they equal?
true
Canary called jOhn vs Rizz are they equal?
false



Ostrich called Maxwell vs Maxwell are they equal?
true
Ostrich called Maxwell vs Ganyu are they equal?
false



Shark called Yanfei vs Yanfei are they equal?
true
Shark called Yanfei vs BOBBLES are they equal?
false



Trout called Ahri vs Ahri are they equal?
true
Trout called Ahri vs Redox are they equal?
false


Checking that equals methods work for different types

Trout called 'Ahri' vs Trout called 'Ahri' are they equal?
true
Trout called 'Ahri' vs Shark called 'Yanfei' are they equal?
false
Ostrich called 'Maxwell' vs Trout called 'Ahri' are they equal?
false
Ostrich called 'Maxwell' vs Canary called 'Rizz' are they equal?
false
```

Canary code

```
public class Canary extends Bird
{

    /**
     * Constructor for objects of class Canary
     */
    public Canary(String name)
    {
        super(); // call the constructor of the superclass Bird
        this.name = name; //overrides the name given in Animal
        colour = "yellow"; // this overrides the value inherited from Bird
    }

    /**
     * Sing method overrides the sing method
     * inherited from superclass Bird
     */
    @Override // good programming practice to use @Override to denote overridden methods
    public void sing(){
        System.out.println("tweet tweet tweet");
    }

    /**
     * toString method returns a String representation of the bird
     * What superclass has Canary inherited this method from? java.lang.Object
     */
    @Override
    public String toString()
```

```
{
    String strng ="";
    strng+= "Canary; ";
    strng+= "\n";
    // Include the fields and attributes inherited
    //from Bird and Animal in the String representation
    strng+= "is a Bird; ";
    strng+= "hasFeathers: ";
    strng+= hasFeathers;
    strng+= "; ";
    strng+= "hasWings: ";
    strng+= hasWings;
    strng+= "; ";
    strng+= "flies: ";
    strng+= flies;
    strng+= "\n";

    strng+= "is an Animal; ";
    strng+= "name: ";
    strng+= name;
    strng+= "; ";
    strng+= "colour: ";
    strng+= colour;
    strng+= "; ";
    strng+= "hasSkin: ";
    strng+= hasSkin;
    strng+= "; ";
    strng+= "breathes: ";
    strng+= breathes;
    strng+= "; ";
```

```
    strng+= "eats: ";
    strng+= eats;
    strng+= "; ";
    strng+= "\n";


    return strng;
}


/**
 * equals method defines how equality is defined between
 * the instances of the Canary class
 * param Object
 * return true or false depending on whether the input object is
 * equal to this Canary object
 *
 * Rules for equality
 * Same name, same colour
 */

@Override
public boolean equals(Object obj){

    //Make sure that the object reference isn't empty. (no object)
    if(obj == null)
    {
        //no object (obj is null) was input to compare this object to
        return false;
    }
```

```
        //Check that the object reference is the same class as this class
        if(obj instanceof Canary)
        {

            //Upcast the object so that checks can be done below
            Canary can = (Canary)obj;

            //Check the rules for equality
            if(this.name.equals(can.getName())
             && this.colour.equals(can.getColour()))
            {
                //the objects are equal
                return true;
            }
        }

        //the objects are not equal
        return false;
    }

}
```

Ostrich code

```
public class Ostrich extends Bird
{
    boolean isTall;
    boolean hasThinLongLegs;

    /**
     * Constructor for objects of class Ostrich
     // */
    public Ostrich(String name)
    {
        super();
        this.name = name; //Overrides value set in Animal.

        isTall = true;
        hasThinLongLegs = true;
        flies = false; //Override value set in Bird. he can't fly :(
    }

    /**
     * toString method returns a String representation of the bird
     * What superclass has Ostrich inherited this method from? java.lang.Object
     */
    @Override
    public String toString()
    {
        String strng = "";
        strng+= "Ostrich; ";
        strng+= "\n";
```

```
strng+= "isTall: ";
strng+= isTall;
strng+= "; ";
strng+= "hasThinLongLegs: ";
strng+= hasThinLongLegs;
strng+= "; ";
strng+= "\n";

strng+= "is a Bird; ";
strng+= "hasFeathers: ";
strng+= hasFeathers;
strng+= "; ";
strng+= "hasWings: ";
strng+= hasWings;
strng+= "; ";
strng+= "flies: ";
strng+= flies;
strng+= "\n";

strng+= "is an Animal; ";
strng+= "name: ";
strng+= name;
strng+= "; ";
strng+= "colour: ";
strng+= colour;
strng+= "; ";
strng+= "hasSkin: ";
strng+= hasSkin;
strng+= "; ";
strng+= "breathes: ";
```

```
    strng+= breathes;
    strng+= "; ";
    strng+= "eats: ";
    strng+= eats;
    strng+= "; ";
    strng+= "\n";

    return strng;
}

/*
 * Equals method for leaf class.
 * This method defines what makes Ostrich objects equal
 * Returns true if the objects are equal, else false.
 *
 * Rules for equality
 * Same name, same isTall, same hasThinLongLegs
 */
@Override
public boolean equals(Object obj)
{
    if(obj == null)
    {
        //no object input
        return false;
    }

    if(obj instanceof Ostrich)
    {
        Ostrich ost = (Ostrich)obj;
```

```
            if(this.name.equals(ost.getName())
             && this.isTall == ost.isTall()
             && this.hasThinLongLegs == ost.hasThinLongLegs())
            {
                //objects equal
                return true;
            }
        }

        //objects not equal
        return false;
    }

    //Getter for isTall attribute
    public boolean isTall()
    {
        return isTall;
    }

    //Getter for hasThinLongLegs attribute
    public boolean hasThinLongLegs()
    {
        return hasThinLongLegs;
    }
}
```

Fish code

```
public abstract class Fish extends Animal
{
    boolean hasFins;
    boolean hasGills;

    /**
     * Constructor for objects of class Fish
     */
    public Fish()
    {
        hasFins = true;
        hasGills = true;
        hasSkin = false; //Override value set in Animal. All subclasses inherit this value.
    }

    @Override
    public void move(int distance)
    {
        System.out.printf("I swim %d metres \n", distance);
    }

    //Getter for hasFins attribute
    public boolean hasFins()
    {
        return hasFins;
    }

    //Getter for hasGills attribute
```

```
    public boolean hasGills()
    {
        return hasGills;
    }
}
```

Shark code

```
public class Shark extends Fish
{
    boolean canBite;
    boolean isDangerous;

    /**
     * Constructor for objects of class Shark
     */
    public Shark(String name)
    {
        super();

        this.name = name; //Overrides value set in Animal.

        canBite = true;
        isDangerous = true;
    }

    /**
     * toString method returns a String representation of the fish
     * What superclass has Shark inherited this method from? java.lang.Object
     */
    @Override
    public String toString()
    {
        String strng = "";

        strng+= "Shark; ";
```

```
strng+= "\n";
strng+= "canBite: ";
strng+= canBite;
strng+= "; ";
strng+= "isDangerous: ";
strng+= isDangerous;
strng+= "; ";
strng+= "\n";

strng+= "is a Fish; ";
strng+= "hasFins: ";
strng+= hasFins;
strng+= "; ";
strng+= "hasGills: ";
strng+= hasGills;
strng+= "; ";
strng+= "\n";

strng+= "is an Animal; ";
strng+= "name: ";
strng+= name;
strng+= "; ";
strng+= "colour: ";
strng+= colour;
strng+= "; ";
strng+= "hasSkin: ";
strng+= hasSkin;
strng+= "; ";
strng+= "breathes: ";
strng+= breathes;
```

```
    strng+= "; ";
    strng+= "eats: ";
    strng+= eats;
    strng+= "; ";
    strng+= "\n";

    return strng;
}

/*
 * Equals method for leaf class.
 * This method defines what makes Shark objects equal
 * Returns true if the objects are equal, else false.
 *
 * Rules for equality
 * Same name, same canBite, same isDangrous
 */
@Override
public boolean equals(Object obj)
{
    if(obj == null)
    {
        //no object input
        return false;
    }

    if(obj instanceof Shark)
    {
        Shark shk = (Shark)obj;
```

```
            if(this.name.equals(shk.getName())
             && this.canBite == shk.canBite()
             && this.isDangerous == shk.isDangerous())
            {
                //objects equal
                return true;
            }
        }

        //objects not equal
        return false;
    }

    //Getter for canBite attribute
    public boolean canBite()
    {
        return canBite;
    }

    //Getter for isDangerous attribute
    public boolean isDangerous()
    {
        return isDangerous;
    }
}
```

Trout code

```
public class Trout extends Fish
{
    boolean hasSpikes;
    boolean isEdible;
    boolean swimsUpriverToLayEggs;

    /**
     * Constructor for objects of class Trout
     */
    public Trout(String name)
    {
        super();

        this.name = name; //Overrides value set in Animal.

        hasSpikes = true;
        isEdible = true;
        swimsUpriverToLayEggs = true;
        colour = "brown"; //Overrides value set in Animal
    }

    /**
     * toString method returns a String representation of the fish
     * What superclass has Trout inherited this method from? java.lang.Object
     */
    @Override
    public String toString()
    {
```

```
String strng = "";

strng+= "Trout; ";
strng+= "\n";
strng+= "hasSpikes: ";
strng+= hasSpikes;
strng+= "; ";
strng+= "isEdible: ";
strng+= isEdible;
strng+= "; ";
strng+= "swimsUpriverToLayEggs: ";
strng+= swimsUpriverToLayEggs;
strng+= "; ";
strng+= "\n";

strng+= "is a Fish; ";
strng+= "hasFins: ";
strng+= hasFins;
strng+= "; ";
strng+= "hasGills: ";
strng+= hasGills;
strng+= "; ";
strng+= "\n";

strng+= "is an Animal; ";
strng+= "name: ";
strng+= name;
strng+= "; ";
strng+= "colour: ";
strng+= colour;
```

```
    strng+= "; ";
    strng+= "hasSkin: ";
    strng+= hasSkin;
    strng+= "; ";
    strng+= "breathes: ";
    strng+= breathes;
    strng+= "; ";
    strng+= "eats: ";
    strng+= eats;
    strng+= "; ";
    strng+= "\n";

    return strng;
}

/*
 * Equals method for leaf class.
 * This method defines what makes Trout objects equal
 * Returns true if the objects are equal, else false.
 *
 * Rules for equality
 * Same name, same hasSpikes, same isEdible, same swimsUpriverToLayEggs
 */
@Override
public boolean equals(Object obj)
{
    if(obj == null)
    {
        //no object input
        return false;
```

```
    }

    if(obj instanceof Trout)
    {
        Trout trt = (Trout)obj;

        if(this.name.equals(trt.getName())
         && this.hasSpikes == trt.hasSpikes()
         && this.isEdible == trt.isEdible()
         && this.swimsUpriverToLayEggs == trt.swimsUpriverToLayEggs())
        {
            //objects equal
            return true;
        }
    }

    //objects not equal
    return false;
}

//Getter for hasSpikes attribute
public boolean hasSpikes()
{
    return hasSpikes;
}

//Getter for isEdible attribute
public boolean isEdible()
{
    return isEdible;
```

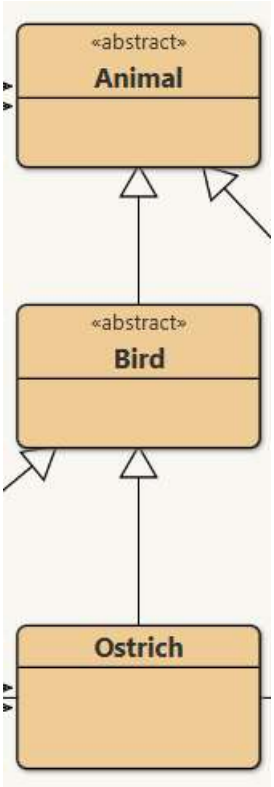
```
    }

    //Getter for swimsUpriverToLayEggs attribute
    public boolean swimsUpriverToLayEggs()
    {
        return swimsUpriverToLayEggs;
    }
}
```

A brief explanation of my code

How did the Ostrich print: “I am a bird but cannot fly.”

A Ostrich object is created. There is an inheritance relationship in place. An Ostrich is a Bird. A Bird is an Animal.

This is possible due to the “extends” keyword.




```
public abstract class Animal
```

```
public abstract class Bird extends Animal
```

```
public class Ostrich extends Bird
```

(The “abstract” keyword just means that the class is not allowed to be instantiated as an object. This is because, as the programmer, I decided that I don’t want any “Bird” objects to ever exist. This just helps me not create unwanted objects by mistake).

We can use an animal reference type to point to our Ostrich object because an Ostrich effectively is-an Animal.

Animal animal = new Ostrich(“Jerry”);

The Ostrich object has all of the properties of Animal:

```
boolean hasSkin;
boolean breathes;
String colour;
boolean eats;

String name;
```

(these are general properties – all animals have them)

And the Ostrich object has all of the properties of Bird:

```
boolean hasFeathers;
boolean hasWings;
boolean flies;
```

(these are general properties – all birds have them)

And the Ostrich object has the properties of Ostrich:

```
boolean isTall;
boolean hasThinLongLegs;
```

So all of these properties are available in the Ostrich object.

The Animal class has an abstract function.

An abstract method is a definition for a method that must be implemented by one its subclasses.

```
public abstract void move(int distance);
```

It's purpose is to define the method signature so that any of the the subclasses can override it to make move do their own thing.

Next page..

The Bird class has a concrete method for move.

```
@Override
public void move(int distance)
{
    //If the Bird cannot fly the output should be different.
    // Check this using an "if" and the "flies" boolean.
    if(flies)
    {
        System.out.printf("I fly %d metres \n", distance);
    }
    else
    {
        System.out.printf("I am a bird but cannot fly.\n");
    }
}
```

WHAT HAPPENS WHEN MOVE IS CALLED ON AN OSTRICH OBJECT.

Animal animal = new Ostrich(“Jerry”);

animal.move(5);

The program looks in the Ostrich class for the move method. It cannot find it. The program looks in the Ostrich’s parent class. It found a move(int distance) method in the Bird class. Cool. It runs that. A simple check is done to see if the Boolean “flies” is true, denoting that the animal can fly. As the ostrich cannot fly, its “flies” variable is false. So it “I am a bird but cannot fly” is printed out.