OOP: Assignment 3

Assignment & code Description:

This assignment is centralised around learning how to use inheritance which involves using abstract classes, for example my animal, Fish and Bird are abstract classes and It also involves combining these classes with concrete classes that inherit there field values from the class that abstracts them and we can change these values within the concrete classes to suit what we want, concrete classes within my program are Shark, Trout which are abstracted by Fish, then Canary, Ostrich which are abstracted by the abstract class Bird. Another goal of this assignment was to code and use a toString and toEquals method, toString is basically just building up a string up of the various fields found within the class and toEquals checks to see if the object is equal to the object that is passed into the method.

Finally, we had to test our work within a main method, and Through the use of an array and a for loop I printed out the various aspects of each concrete class using System.out.println(animals[i]) rather than calling the toString method directly. Then to test the toEquals I used if statements along with or operators to compare the different objects. Note:(I Believe there is a better way to achieve this using a for loop also I wondered how I could print just the names of the animal, but anytime I tried this it either printed the entire toString output or gave me an error when I tried to use animals[0].name) Main is further outlined below and I have highlighted the comments in green then I highlighted the different titles in yellow, just to make them stand out more to whoever is correcting this.

Each Concrete class contains a toString and a toEquals these are outlined within the comments in my code. I made a few key design choices in my shark and Fish class, I placed a lot of the fields from shark and trout into Fish as I felt that they applied to Fish in General for example isEdible and isDangerous are in Fish because you could apply these Booleans to basically every Fish in the world. Trout and Ostrich have some local variables like isBrown, RiverReproduction, isTall because I feel they could be quite redundant if they were placed in the abstract classes.

Fish and Bird contain the majority of the fields and these will be inherited by the concrete classes, but it also inherits the move method and a couple of fields from the animal abstract class, also there are getter methods for each field so there Boolean value can be accessed later. The move method within the Bird and Fish class play an important role i.e. not every Bird can fly and not every fish swims(although this is rare) so I use the flies and the canSwim Booleans to check if this instance of Ostrich for example, flies and this is clearly false so it will print how far the ostrich walked rather than flied.

Animal is fairly straight forward it contains a couple of very basic fields and the basis and declaration of our move method.

AnimalTest code:

In this main class I have three tests, the first uses a for loop to loop through an array of the various animal objects and prints them out as well as their move method.

Test 2 is the first part of my toEquals test where I show if there were no objects the same what the output would be. I use if statements to see if the objects are equal.

OOP: Assignment 3

In Test 3 I Show when two objects are the same and I print out the position in the array the animal object is because if I do it by printing the animal object itself it prints the big block from the toString method, this is the second part of my test 2 essentially.

```
public class Main

{

   public static void Main(String[] args)

   {

      test1();

      test2();

      test3();

   }

   private static void test1()

      {

      Animal[] animals = new Animal[4]; //Declare Array of size 4

      animals[0] = new Canary("JFK"); // fill array with object references

      animals[1] = new Trout("Benji");

      animals[2] = new Ostrich("Lambo");

      animals[3] = new Shark("Lorry");

         for(Animal animal: animals){ // loop through animals

          System.out.println(animal); // print out the animal which will automatically call the toString

         animal.move(20); // each animal will get different outputs from move because of how i've set
it up within Bird and Fish

         System.out.println("``````````````````````````````````````````````````````````````````````````````````````````````````");

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


      }


      }


   private static void test2()

   {

      Animal[] animals = new Animal[4]; //Declare array
```

```
        animals[0] = new Canary("JFK"); //fill array with animal object references

        animals[1] = new Trout("Benji");

        animals[2] = new Ostrich("Lambo");

        animals[3] = new Shark("Lorry");


        System.out.println("---Test 2---");

          if(animals[0].equals(animals[1]) || animals[0].equals(animals[2]) ||
animals[0].equals(animals[3])){

            System.out.println("animals[0] is equal to one of the objects");

           } // compare each animal to every other one and print out if its equal to one or not

          else{

           System.out.println("animals[0] is not equal to one of the objects");

          }

          if(animals[1].equals(animals[0]) || animals[1].equals(animals[2]) ||
animals[1].equals(animals[3])){

            System.out.println("animals[1] is equal to one of the objects");

          }

          else{

            System.out.println("animals[1] is not equal to one of the objects ");

          }

          if(animals[2].equals(animals[0]) || animals[2].equals(animals[1])||
animals[2].equals(animals[3])){

            System.out.println("animals[2] is equal to one of the objects");

          }

          else{

            System.out.println(" animals[2] is not equal to one of the objects");

          }

          if(animals[3].equals(animals[0])  || animals[3].equals(animals[1])||
animals[3].equals(animals[2])){

            System.out.println("animals[3] is equal to one of the objects");

          }

          else{
            System.out.println("animals[3] is not equal to one of the objects");
          }



        System.out.println("``````````````````````````````````````````````````````````````````````````````````````````````````````");



    }


    private static void test3(){
        Animal[] animals = new Animal[5]; //Declare array of animals
        animals[0] = new Canary("JFK");
        animals[1] = new Canary("JFK"); //Fill array with animal object Refernces
        animals[2] = new Shark("Lorry");
        animals[3] = new Shark("Lorry");
        animals[4] = new Trout("Marky Mark");
        // in this example their will be two objects equal to each other
        System.out.println("---Test 3---");


          if(animals[0].equals(animals[1]) || animals[0].equals(animals[2]) ||
animals[0].equals(animals[3]) || animals[0].equals(animals[4])){
            System.out.println("animals[0] is equal to one of the objects");
          } // compare each animal to every other one and print out if its equal to one or not
          else{
            System.out.println("animals[0] is not equal to one of the objects");
          }
          if(animals[1].equals(animals[0]) || animals[1].equals(animals[2]) ||
animals[1].equals(animals[3]) || animals[1].equals(animals[4])){
            System.out.println("animals[1] is equal to one of the objects");
          }
          else{
            System.out.println("animals[1] is not equal to one of the objects ");
```

```
        }

        if(animals[2].equals(animals[0]) || animals[2].equals(animals[1])||
animals[2].equals(animals[3])|| animals[2].equals(animals[4])){

          System.out.println("animals[2] is equal to one of the objects");

        }

        else{

          System.out.println("animals[2] is not equal to one of the objects");

        }

        if(animals[3].equals(animals[0])  || animals[3].equals(animals[1])||
animals[3].equals(animals[2])|| animals[3].equals(animals[4])){

          System.out.println("animals[3] is equal to one of the objects");

        }

        else{

          System.out.println("animals[3] is not equal to one of the objects");

        }

        if(animals[4].equals(animals[0])  || animals[4].equals(animals[1])||
animals[4].equals(animals[2])|| animals[4].equals(animals[3])){

          System.out.println("animals[4] is equal to one of the objects ");

        }

        else{

          System.out.println("animals[4] is not equal to one of the objects");

        }

    }

}
```

Test1 Output:

```
Canary:

name?
```

OOP: Assignment 3

~~~
JFK

colour?

yellow

Does this bird fly?

true

Does this bird have feathers?

true


Does this bird have wings?

true

I fly 20 metres
``````````````````````````````````````````````````````````````````````````````````````



Trout:

name?

Benji

The colour of this Trout is?

Rainbow

Does this Trout have Fins?

true

Does this Trout have Gills?

true

Can this Trout Swim?

true

Is this Trout dangerous?

false

Is this Trout brown?

false

Does this Trout have spikes?

true
~~~

OOP: Assignment 3

~~~~
Does this trout reproduce up river?

true

I swam 20 metres

`````````````````````````````````````````````````````````````````````````````````````````````````



Ostrich:

name?

Lambo

colour?

Black & Grey

Does this bird fly?

false

Does this bird have feathers?

true

Does this bird have wings?

true

Is this bird tall?

true

I am a bird but cannot fly. I walked 20 metres


`````````````````````````````````````````````````````````````````````````````````````````````````



Shark:

name:

Lorry

colour:

Grey

Does this Shark have fins?

true
~~~~

Does this shark have Gills

true

Can this shark Swim?

true

Is this shark dangerous?

true

Is this shark edible?

false

I swam 20 metres

````````````````````````````````````````````````````````````````````````````````````````````

## Test2 Output:

---Test 2---

animals[0] is not equal to one of the objects

animals[1] is not equal to one of the objects

 animals[2] is not equal to one of the objects

animals[3] is not equal to one of the objects

````````````````````````````````````````````````````````````````````````````````````````````

Test3 Output:

---Test 3---

animals[0] is equal to one of the objects

animals[1] is equal to one of the objects

animals[2] is equal to one of the objects

animals[3] is equal to one of the objects

animals[4] is not equal to one of the objects

Canary Code:

```
public class Canary extends Bird
```

```
{
  String name;  // the name of this Canary

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



  @Override //overridden sing method from bird superclass
   public void sing(){
    System.out.println("tweet tweet tweet");
  }

  /**
   * toString method returns a String representation of the bird
   */
  @Override
  public String toString(){
    String strng ="";
    strng+= "Canary: ";
    strng+= "\n";
    strng+= "name? ";
    strng+= "\n";
    strng+= name;
    strng+= "\n";
    strng+= "colour? ";
    strng+= "\n";
    strng+= colour;
    strng+= "\n";
    strng+= "Does this bird fly? ";
    strng+= "\n";
    strng+= flies;
    strng+= "\n";
    strng+= "Does this bird have feathers? ";
    strng+= "\n";
    strng+= hasFeathers(); // use the getter methods to return true or false and print that out.
    strng+= "\n";
    strng+= "\n";
    strng+= "Does this bird have wings? ";
    strng+= "\n";
    strng+= hasWings();

    return strng;
  }




  @Override
  public boolean equals(java.lang.Object object){
    if(this == object){// if the object is being compared to itself return true
      return true;
    }
      if(object instanceof Canary){//if the object is a Canary object
        Canary canary = (Canary) object;// cast as a canary object
        if(name ==  canary.name && colour == canary.colour){// if the colour & name are the same
its the same object thus return true
```

```
            return true;
          }
        }
        return false; // if its not equal return false
      }


}
```

Ostrich Code:

```
public class Ostrich extends Bird
{
    String name;
    boolean isTall; // I put isTall in Ostrich because it dosent apply to many birds
    public Ostrich(String name)
    {
        super();  // call the constructor of the superclass Bird
        this.name = name;
        colour = "Black & Grey";  // this overrides the values inherited from Bird
        flies = false;
        isTall = true;
    }

    @Override
    public void sing(){ // an overrridden method from bird
        System.out.println("Bwak Bwak Bwak");
    }

    /**
     * toString method returns a String representation of the Ostrich
     */
```

```
@Override
public String toString(){ //toString method to print the fields that pertain to this Ostrich
   String strng ="";
   strng+= "Ostrich: ";
   strng+= "\n";
   strng+= "name? ";
   strng+= "\n";
   strng+= name;
   strng+= "\n";
   strng+= "colour? ";
   strng+= "\n";
   strng+= colour;
   strng+= "\n";
   strng+= "Does this bird fly? ";
   strng+= "\n";
   strng+= flies;
   strng+= "\n";
   strng+= "Does this bird have feathers? ";
   strng+= "\n";
   strng+= hasFeathers(); // use the getter methods to return true or false and print that out.
   strng+= "\n";
   strng+= "Does this bird have wings? ";
   strng+= "\n";
   strng+=  hasWings();
   strng+= "\n";
   strng+= "Is this bird tall? ";
   strng+= "\n";
   strng+= isTall;
   return strng;
}
```

```
    @Override
    public boolean equals(java.lang.Object object){

        if(this == object){ // if the object is being compared to itself return true
            return true;
        }
            if(object instanceof Ostrich){ //if the object is a Ostrich object
                Ostrich ostrich = (Ostrich) object; //cast as a Ostrich
                if(name ==  ostrich.name && colour == ostrich.colour){ // if the colour & name are the same
its the same object thus return true
                return true;
            }
        }
        return false; //if not equal return false
    }

}
```

Fish Code:

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

    public Fish()
    {
```

```
    super();  //calls the constructor of superclass  - Animal

    colour = "black";  //overrides the value of colour inherited from Animal

    hasFins = true;  //all the subclasses of Fish inherit this property and value

    canSwim = true;

    hasGills = true;

    isDangerous = true;

    isEdible = true;


  }


  /**

   * move method overrides the move method

   * inherited from superclass Animal

   */

  @Override

  public void move(int distance){

    if(canSwim){ //check if the fish can swim, if it can print the first line else print its a fish that
doesnt swim.

      System.out.printf("I swam %d metres \n", distance);

     }else{

      System.out.printf("I am a Fish but I do not Swim. I crawled %d metres \n\n", distance);

    }

  }


  //method bite will be inherited by subclasses of Fish and can be overidden

  public void bite(){

     System.out.println("Munch Munch"); // Just a simple mehtod that prints out a generic bite
noise

  }


  public boolean hasGills(){ // getter method for hasGills

    return hasGills;
  }

  public boolean hasFins(){ //getter method for hasFins
    return hasFins;
  }

  public boolean canSwim(){ //getter method for canSwim
    return canSwim;
  }

  public boolean isDangerous(){ //getter method for isDangerous
  return isDangerous;
  }

  public boolean isEdible(){ //getter method for isEdible
  return isEdible;
  }
  // all subclasses inherit these getter methods
}
```

Shark Code:

```
public class Shark extends Fish
{
  String name;
  public Shark(String name)
   {
     super(); // call the constructor of the superclass Fish
    this.name = name;
    colour = "Grey"; // this overrides the value inherited from Fish
    isDangerous = true;

    isEdible = false;

}



@Override// my toString method which prints out the various fields to do with this shark

 public String toString(){

    String strng ="";

    strng+= "Shark: ";

    strng+= "\n";

    strng+= "name: ";

    strng+= "\n";

    strng+= name;

    strng+= "\n";

    strng+= "colour: ";

    strng+= "\n";

    strng+= colour;

    strng+= "\n";

    strng+= "Does this Shark have fins? ";

    strng+= "\n";

    strng+= hasFins(); // use the getter methods to return true or false and print that out.

    strng+= "\n";

    strng+= "Does this shark have Gills";

    strng+= "\n";

    strng+=  hasGills();

    strng+= "\n";

    strng+= "Can this shark Swim? ";

    strng+= "\n";

    strng+= canSwim();

    strng+= "\n";

    strng+= "Is this shark dangerous? ";
```

```
        strng+= "\n";

        strng+= isDangerous();

        strng+= "\n";

        strng+= "Is this shark edible? ";

        strng+= "\n";

        strng+= isEdible();

        return strng; // return the entire string

    }


     @Override

    public boolean equals(java.lang.Object object){

        if(this == object){

            return true; // if the object is being compared to itself return true

        }

            if(object instanceof Shark){  //if the object is a shark object

                Shark shark = (Shark) object; //cast it as a shark object

                if(name ==  shark.name && colour == shark.colour){ // if the colour & name are the same its
the same object thus return true

                return true;

            }

        }

        return false; //return false if it is not equal to the object

    }


    @Override

    public void bite(){ //An overridden method from Fish

    System.out.println("Chomp Chomp");

    }


}
```

OOP: Assignment 3

Trout Code:

```
public class Trout extends Fish

{

   String name;

  boolean isBrown; // I put these three fields in trout specifically because they pertain to trout more
than Fish in general

  boolean riverReproduction;

  boolean hasSpikes;

  public Trout(String name)

  {

    super(); // call the constructor of the superclass Fish

    this.name = name;

    colour = "Rainbow"; // this overrides the values inherited from Fish

    isDangerous = false;

    isBrown = true;

    hasSpikes = true;

    riverReproduction = true;

  }



   @Override// my toString method which prints out the various fields to do with this trout

   public String toString(){

     String strng ="";

     strng+= "Trout: ";

     strng+= "\n";

     strng+= "name? ";

     strng+= "\n";

     strng+=  name;

     strng+= "\n";

     strng+= "The colour of this Trout is? ";

     strng+= "\n";
```

```
strng+=  colour;

strng+= "\n";

strng+= "Does this Trout have Fins? ";

strng+= "\n";

strng+=  hasFins();

strng+= "\n";

strng+= "Does this Trout have Gills? ";

strng+= "\n";

strng+=  hasGills();

strng+= "\n";

strng+= "Can this Trout Swim? ";

strng+= "\n";

strng+=  canSwim();

strng+= "\n";

strng+= "Is this Trout dangerous? ";

strng+= "\n";

strng+=  isDangerous(); // use the getter methods to return true or false and print that out.

strng+= "\n";

strng+= "Is this Trout brown? ";

strng+= "\n";

strng+=  getIsBrown();

strng+= "\n";

strng+= "Does this Trout have spikes? ";

strng+= "\n";

strng+=  hasSpikes;

strng+= "\n";

strng+= "Does this trout reproduce up river? ";

strng+= "\n";

strng+=  riverReproduction;


return strng; // return the final string
```

```
 }

 @Override
 public boolean equals(java.lang.Object object){

    if(this == object){ // if the object is being compared to itself return true
        return true;
    }
       if(object instanceof Trout){ //if the object is a Trout object
          Trout trout = (Trout) object;  //cast it as a Trout object
          if(name ==  trout.name){ // if the colour & name are the same its the same object thus
return true
          return true;
       }
    }
    return false; //if its not equal return false
 }

 @Override
public void bite(){ // method that is overridding the method from its parent class Fish
 System.out.println("Nibble Nibble");
 }

public boolean getIsBrown(){ //simple method to check whether the trout is brown or not
 if(colour == "brown"){
    return isBrown = true;
 }
 else{
    return isBrown = false;
 }
}
```

```
}
```