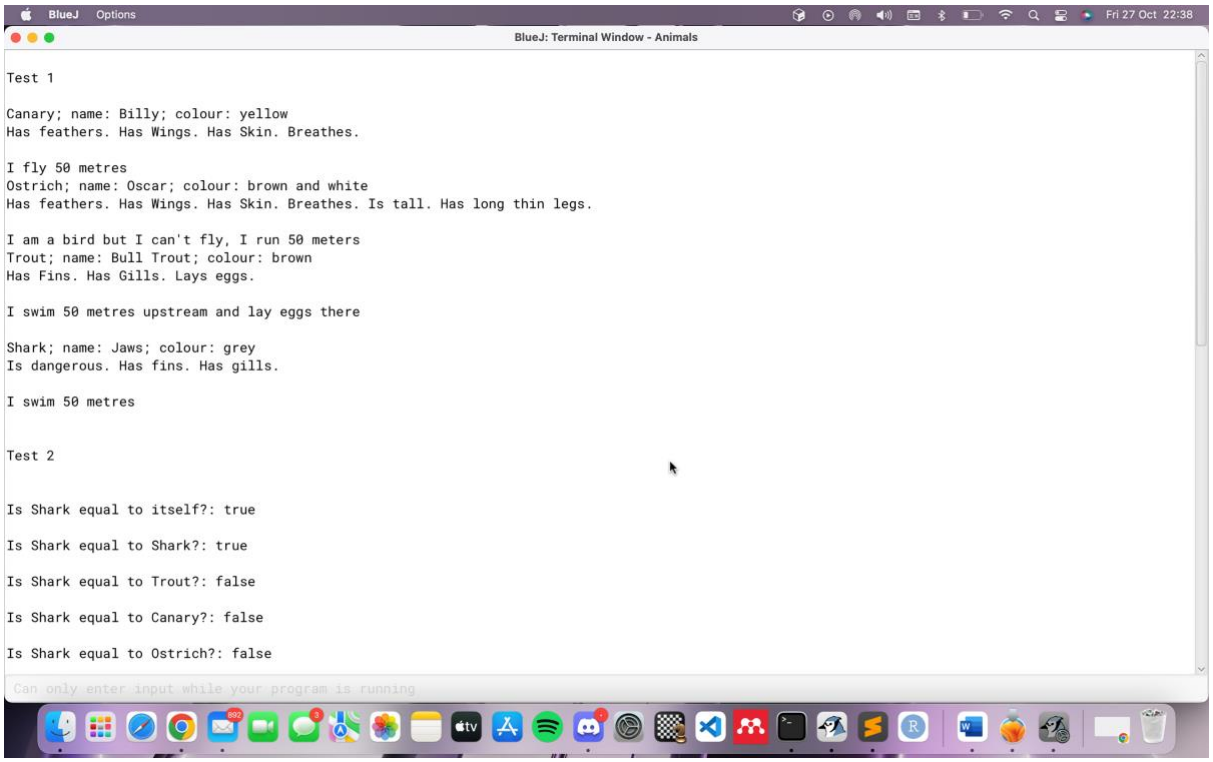
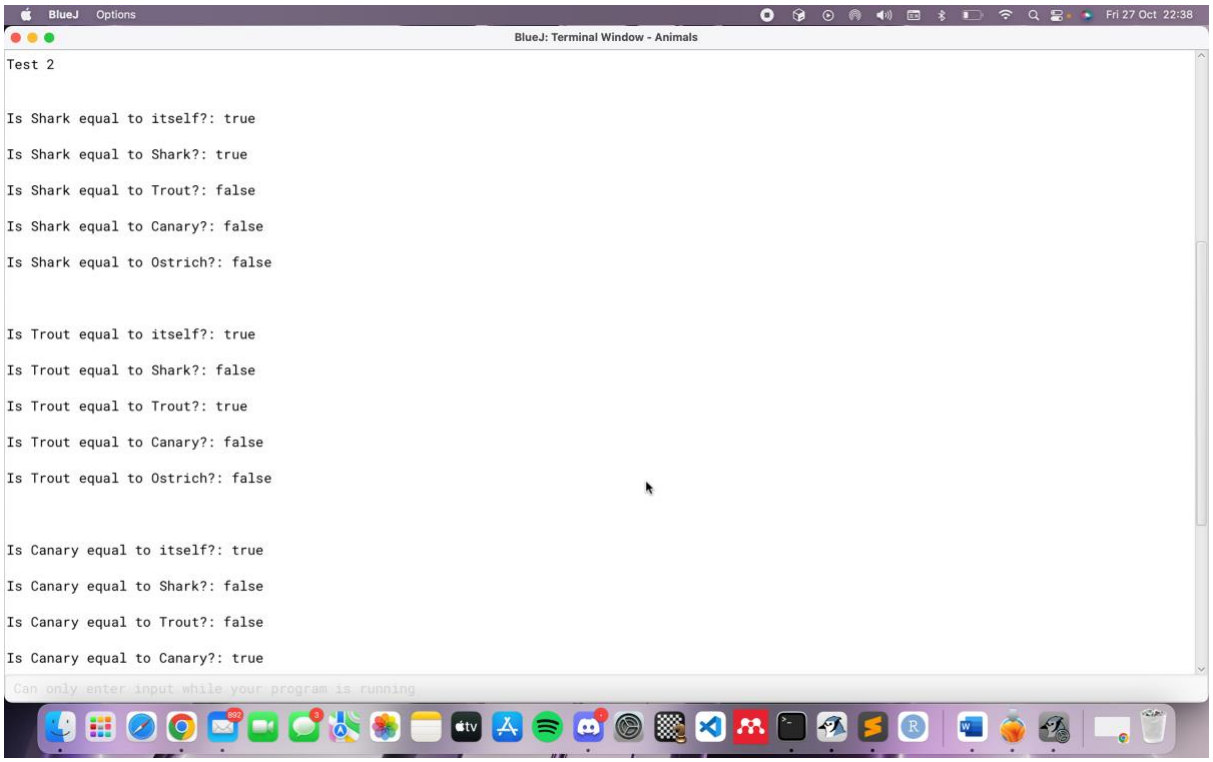
OOP assignment 3
Ciaran Gray 22427722

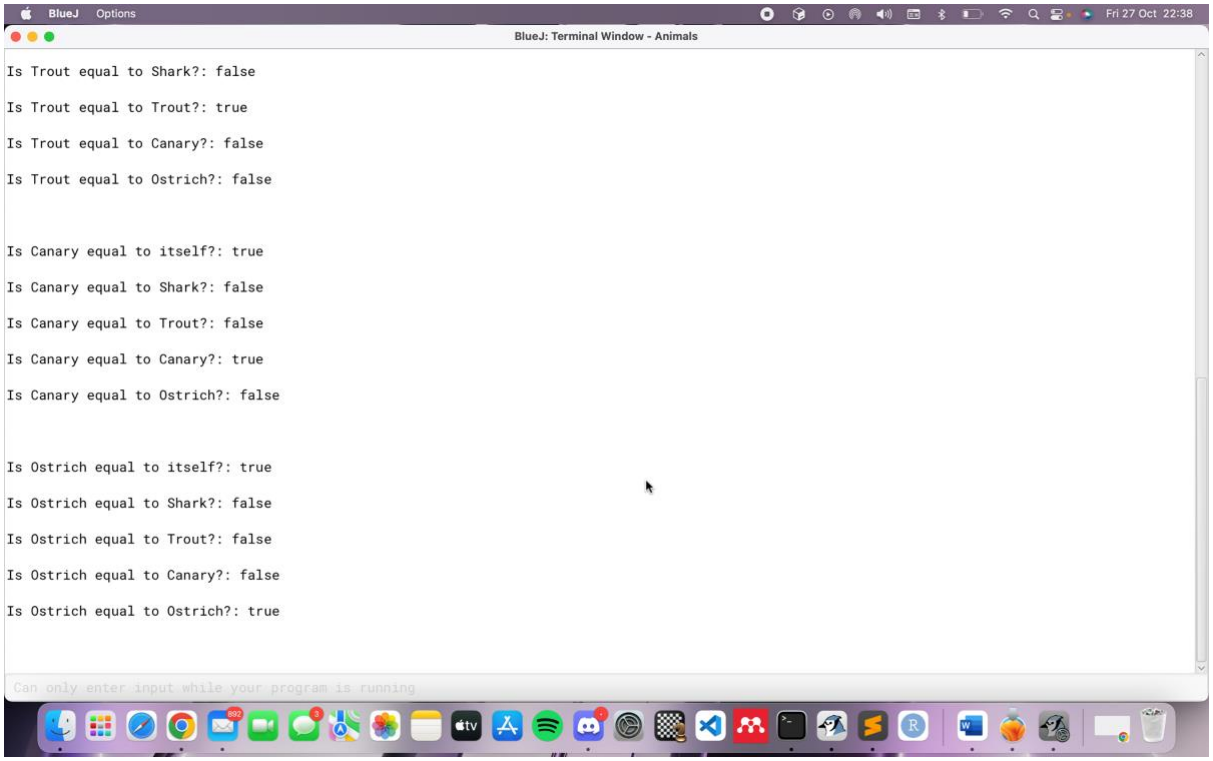
1.
For this code I already have the code for the parent classes animal and bird. I have the canary class set up, so all I need to do is implement a toString and equals method for this class and then repeat the process for the ostrich class, and then create another animal subclass – fish, and then 2 more fish subclasses shark and trout which will follow the same structure and hierarchy. For the toString methods I'll have to just add in if staements that will add strings containing certain characteristics if they apply, for example, if a bird can fly, i.e if the "flies" Boolean is set to true for the canary, it will print out that the bird, for the ostrich, it wont print out anything as flies will be set to false. For the equals method, ill have to create a new object of the class, first check that it's not null, and then check If the object is of the same class type of that certain class, and then finally I will check the object against one of the classes characteristics, colour, for example. The subclasses constructers will all include any of the fields from the parent's classes that need to be changed and also the super() method. Then the Animaltest class will create an array of animals and add in a new animal object at each index, canary, ostrich, trout and shark, and then loop through the entire array and for test1, print out each class, which will then trigger the toString method and print out the details of each animal, and then for test2 after each iteration of the loop it'll call the equals method and compare the class against itself and every other class.

2.
```
public class AnimalTest
{
   public static void main (String args[]) {
      AnimalTest test = new AnimalTest(); //create a new test object
      test.test1(); //test 1 and test 2 are called
      test.test2();
   }

   public void test1 () {
      System.out.println("\nTest 1\n");

      //create array of animal objects of length 4
      Animal[] animals = new Animal[4];

      //create new animal objects
      Shark shark = new Shark("Jaws");
      Trout trout = new Trout("Bull Trout");
      Canary canary = new Canary("Billy");
      Ostrich ostrich = new Ostrich("Oscar");

      //assign each array slot to each class
      animals[0] = canary;
```

```
        animals[1] = ostrich;
        animals[2] = trout;
        animals[3] = shark;

        //print each animal's toString method and its move method
        for (Animal animal : animals) {
           System.out.println(animal);
           animal.move(50);
        }
     }

        public void test2() {
        System.out.println("\nTest 2\n");

        //initialise the array
        Animal[] animals = new Animal[4];

        //create the animal objects
        Shark shark = new Shark("Jaws");
        Trout trout = new Trout("Bull Trout");
        Canary canary = new Canary("Billy");
        Ostrich ostrich = new Ostrich("Oscar");


        //assign each array slot to each class
        animals[0] = shark;
        animals[1] = trout;
        animals[2] = canary;
        animals[3] = ostrich;


        //for each animal, check if it is equal to itself and every animal
        for (Animal animal : animals) {
           System.out.println("\nIs "+animal.getClass().getSimpleName()+" equal to itself?:
"+animal.equals(animal));
           System.out.println("\nIs "+animal.getClass().getSimpleName()+" equal to Shark?:
"+animal.equals(shark));
           System.out.println("\nIs "+animal.getClass().getSimpleName()+" equal to Trout?:
"+animal.equals(trout));
           System.out.println("\nIs "+animal.getClass().getSimpleName()+" equal to Canary?:
"+animal.equals(canary));
           System.out.println("\nIs "+animal.getClass().getSimpleName()+" equal to Ostrich?:
"+animal.equals(ostrich));
           System.out.println("\n");
        }
     }
}
```


BlueJ Options
Fri 27 Oct 22:38
BlueJ: Terminal Window - Animals
Test 1
Canary; name: Billy; colour: yellow
Has feathers. Has Wings. Has Skin. Breathes.
I fly 50 metres
Ostrich; name: Oscar; colour: brown and white
Has feathers. Has Wings. Has Skin. Breathes. Is tall. Has long thin legs.
I am a bird but I can't fly, I run 50 meters
Trout; name: Bull Trout; colour: brown
Has Fins. Has Gills. Lays eggs.
I swim 50 metres upstream and lay eggs there
Shark; name: Jaws; colour: grey
Is dangerous. Has fins. Has gills.
I swim 50 metres
Test 2
Is Shark equal to itself?: true
Is Shark equal to Shark?: true
Is Shark equal to Trout?: false
Is Shark equal to Canary?: false
Is Shark equal to Ostrich?: false
Can only enter input while your program is running


BlueJ Options
Fri 27 Oct 22:38
BlueJ: Terminal Window - Animals
Test 2
Is Shark equal to itself?: true
Is Shark equal to Shark?: true
Is Shark equal to Trout?: false
Is Shark equal to Canary?: false
Is Shark equal to Ostrich?: false
Is Trout equal to itself?: true
Is Trout equal to Shark?: false
Is Trout equal to Trout?: true
Is Trout equal to Canary?: false
Is Trout equal to Ostrich?: false
Is Canary equal to itself?: true
Is Canary equal to Shark?: false
Is Canary equal to Trout?: false
Is Canary equal to Canary?: true
Can only enter input while your program is running




```
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

   }

   /**
    * 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?
     */
    @Override
    public String toString(){ // certain properties will add on to the toString method if the
canary has them
        String strng ="";
        strng+= "Canary; ";
        strng+= "name: ";
        strng+= name;
        strng+= "; ";
        strng+= "colour: ";
        strng+= colour;
        strng+= "\n";
        if (hasFeathers) {
            strng += "Has feathers. ";
        }
        if (hasWings) {
            strng += "Has Wings. ";
        }
        if (hasSkin) {
            strng += "Has Skin. ";
        }
        if (breathes) {
            strng += "Breathes. ";
        }
        if (isTall) {
            strng += "Is tall. ";
        }
        if (hasLongThinLegs) {
            strng += "Has long thin legs. "; // for example, then canary doesn't have long thin legs,
so it wont be included in it's toString method
        }
        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
     */
```

```
   @Override
   public boolean equals(java.lang.Object obj){
      //his line creates a new Canary object named 'canary' for attribute comparison
      Canary canary = new Canary("Billy");
      //first comparision avoids null pointer exception
      //the second comparision checks if the "obj" object is the same class type of "Canary"
      //the last comparision compares the 'colour' property of this Canary with 'colour' of the
input 'obj'
      if (obj != null && getClass() == obj.getClass() && colour == canary.getColour()) {
         return true;
      }
      return false; //default equals
   }
}

public class Ostrich extends Bird
{
   // instance variables - replace the example below with your own
   String name;

   /**
    * Constructor for objects of class Ostrich
    */
   public Ostrich(String name)
   {
      // initialise instance variables
      super(); // call the constructor of the superclass Bird
      this.name = name;
      colour = "brown and white";// this overrides the value inherited from Bird
      isTall = true;
      flies = false;
      hasLongThinLegs = true;
   }

      @Override // good programming practice to use @Override to denote overridden
methods
   public void sing(){
      System.out.println("Screech!");
   }

   @Override
   public String toString(){
      String strng ="";
      strng+= "Ostrich; ";
      strng+= "name: ";
      strng+= name;
      strng+= "; ";
```

```
        strng+= "colour: ";
        strng+= colour;
        strng+= "\n";
        if (hasFeathers) { //will add bird properties to the string if appropiate
            strng += "Has feathers. ";
        }
        if (hasWings) {
            strng += "Has Wings. ";
        }
        if (hasSkin) {
            strng += "Has Skin. ";
        }
        if (breathes) {
            strng += "Breathes. ";
        }
        if (isTall) {
            strng += "Is tall. ";
        }
        if (hasLongThinLegs) {
            strng += "Has long thin legs. ";
        }
        strng += "\n";
        return strng;
    }

    @Override
    public boolean equals(java.lang.Object obj){
        //his line creates a new Ostrich object named 'ostrich' for attribute comparison
        Ostrich ostrich = new Ostrich("Oscar");
        //first comparision avoids null pointer exception
        //the second comparision checks if the "obj" object is the same class type of "Ostrich"
        //the last comparision compares the 'colour' property of this Ostrich with 'colour' of the
input 'obj'

        if (obj != null && getClass() == obj.getClass() && colour == ostrich.getColour()) {
            return true;
        }
        return false; //default equals
    }
}


public class Fish extends Animal
{
    boolean hasGills; // all subclasses of fish will inherit these fields
    boolean hasFins;
    boolean swims;
```

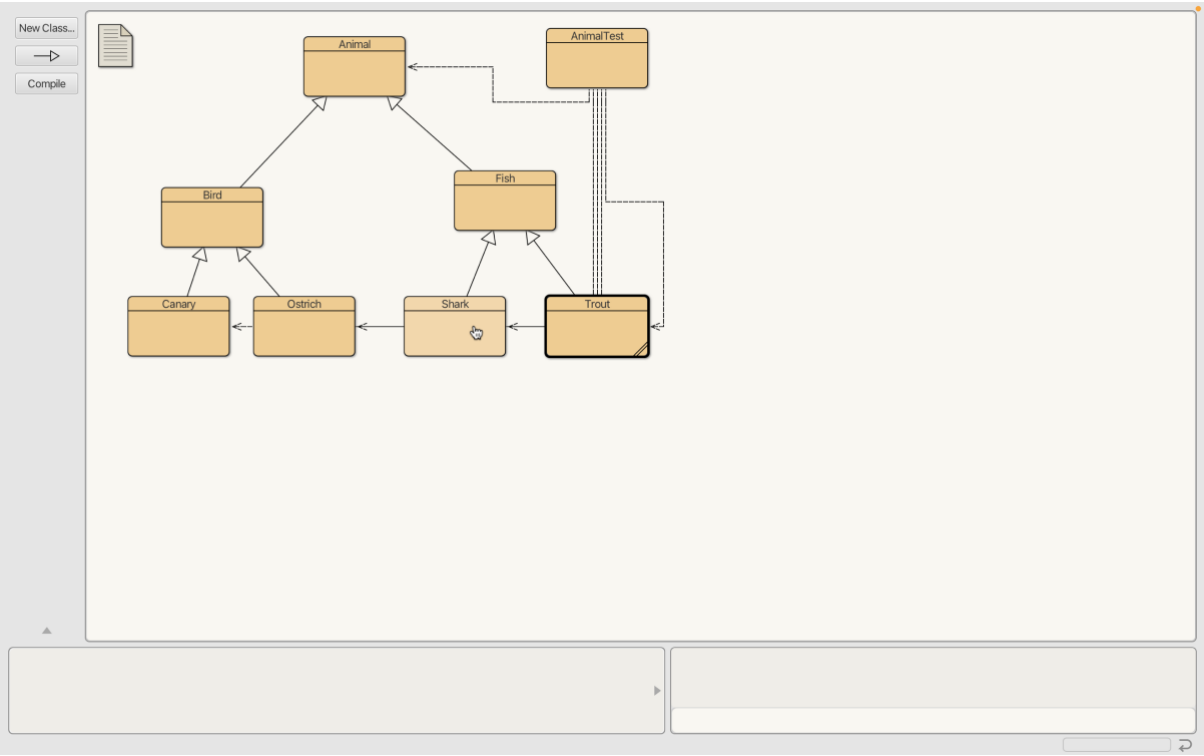
```
    boolean hasSpikes;
    boolean isEdible;
    boolean laysEggs;
    boolean dangerous;
    /**
     * Constructor for objects of class Fish
     */
    public Fish()
    {
        super(); //calls the constructor of its superclass  - Animal
        colour = "black"; //overrides the value of colour inherited from Animal
        hasGills = true; //sets the value of these fields
        hasFins = true;
        swims = true;
        dangerous = false;
        hasSkin = false;
        laysEggs = true;
    }


    @Override // good programming practice to use @Override to denote overridden
methods
    public void move(int distance){
        if (laysEggs && swims) { //the trout swims upstream while the shark doesn't
            System.out.printf("I swim %d metres upstream and lay eggs there \n\n", distance);
        }else if (swims) { //in this case, the shark only swims, he doesn't swim upstream
            System.out.printf("I swim %d metres \n\n", distance);
        }
    }

    //getter methods for all the Fish properties and fields
     public boolean hasGills(){
        return hasGills;
    }

    public boolean hasFins(){
        return hasFins;
    }

    public boolean swims() {
        return swims;
    }

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

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

    public boolean laysEggs() {
        return laysEggs;
    }

    public boolean isDangerous() {
        return dangerous;
    }
}

public class Shark extends Fish
{
    // instance variables - replace the example below with your own
    String name;

    /**
     * Constructor for objects of class Shark
     */
    public Shark(String name)
    {
        // initialise instance variables
        super(); // call the constructor of the superclass Fish
        colour = "grey";// this overrides the value inherited from Fish
        this.name = name;
        dangerous = true;
        laysEggs = false;
    }

    /**
     * An example of a method - replace this comment with your own
     *
     * @param  y  a sample parameter for a method
     * @return    the sum of x and y
     */
    @Override
    public String toString() {
        String strng = "";
        strng += "Shark; ";
        strng += "name: ";
        strng += name;
        strng += "; ";
        strng += "colour: ";
        strng += colour;
        strng += "\n";
```

```
        if (isEdible) { //will add fish properties to the string if appropiate
            strng +="Is edible. ";
        }
        if (dangerous) {
            strng +="Is dangerous. ";
        }
        if (hasSkin) {
            strng +="Has skin. ";
        }
        if (hasFins) {
            strng +="Has fins. ";
        }
        if (hasSpikes){
            strng +="Has spikes. ";
        }
        if (hasGills) {
            strng +="Has gills. ";
        }
        if (laysEggs) {
            strng += "Lays eggs. ";
        }
        strng += "\n";
        return strng;
    }

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

        //his line creates a new Shark object named 'shark' for attribute comparison
        Shark shark = new Shark("Jaws");
        //first comparision avoids null pointer exception
        //the second comparision checks if the "obj" object is the same class type of "Shark"
        //the last comparision compares the 'colour' property of this Shark with 'colour' of the
input 'obj'

        if (obj != null && getClass() == obj.getClass() && colour == shark.getColour()) {
            return true;
        }
        //Otherwise return false
        return false;
    }
}

public class Trout extends Fish
{
    // instance variables - replace the example below with your own
    String name;
```

```
/**
 * Constructor for objects of class Trout
 */
public Trout(String name)
{
   // initialise instance variables
   super(); // call the constructor of the superclass Fish
   this.name = name;
   colour = "brown"; // this overrides the value inherited from Fish
}

/**
 * An example of a method - replace this comment with your own
 *
 * @param  y  a sample parameter for a method
 * @return    the sum of x and y
 */
@Override
public String toString(){ //will add fish properties to the string if appropiate
   String strng ="";
   strng+= "Trout; ";
   strng+= "name: ";
   strng+= name;
   strng+= "; ";
   strng+= "colour: ";
   strng+= colour;
   strng += "\n";
   if (isEdible) {
      strng+="Is Edible. ";
   }
   if (dangerous) {
      strng+="Is Dangerous. ";
   }
   if (hasSkin) {
      strng+="Has Skin. ";
   }
   if (hasFins) {
      strng+="Has Fins. ";
   }
   if (hasSpikes){
      strng+="Has Spikes. ";
   }
   if (hasGills) {
      strng+="Has Gills. ";
   }
   if (laysEggs) {
```

```
            strng += "Lays eggs. ";
        }
        strng += "\n";
        return strng;
    }

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

        //his line creates a new Trout object named 'trout' for attribute comparison
        Trout trout = new Trout("Bull Trout");
        //first comparision avoids null pointer exception
        //the second comparision checks if the "obj" object is the same class type of "Trout"
        //the last comparision compares the 'colour' property of this Trout with 'colour' of the
input 'obj'

        if (obj != null && getClass() == obj.getClass() && colour == trout.getColour()) {
            return true;
        }
        //Otherwise return false
        return false;
    }
}
```



The code works because every subclass gains attributes from it;’s parent classes, for example, ostrich and canary have the same fields and methods but the fields are changed in the actual subclasses depending on the context, every bird subclass has the flies Boolean

variable but it's set to false in ostrich. I changed the moves() method in bird and added an if statement so if flies Boolean is set to false it prints out an alternative message saying that the bird cannot fly, and it walks instead. If flies is true it will print out the standard "I fly x meters" message.