CT2106 OOP: Assignment 3

Inheritance

Assignment Description:

This assignment revolves heavily around how classes inherit methods and variables from each other. The main concept revolves around a hierarchy. The main class, animal, which every single subclass will inherit from is the main class. No matter what type of animal an animal is, be it a bird or fish, it will always belong to the animal class and will inherit its specific variables and methods.

Animals belong to a certain group of animals. Classes bird and fish, group animals of the same type together. Animals of the same group share similar characters that they may not share with other animals outside their group. Each member of this group will inherit characters and methods that are associated with each animal, but also from their group.

Each leaf class represents animals in the hierarchy. Each leaf class has its own special characteristics and/or methods. All birds fly but there are certain exceptions. An ostrich will inherit the fact that it can fly from its group class, but it will have to modify this particular fact to make it more representative of its own situation

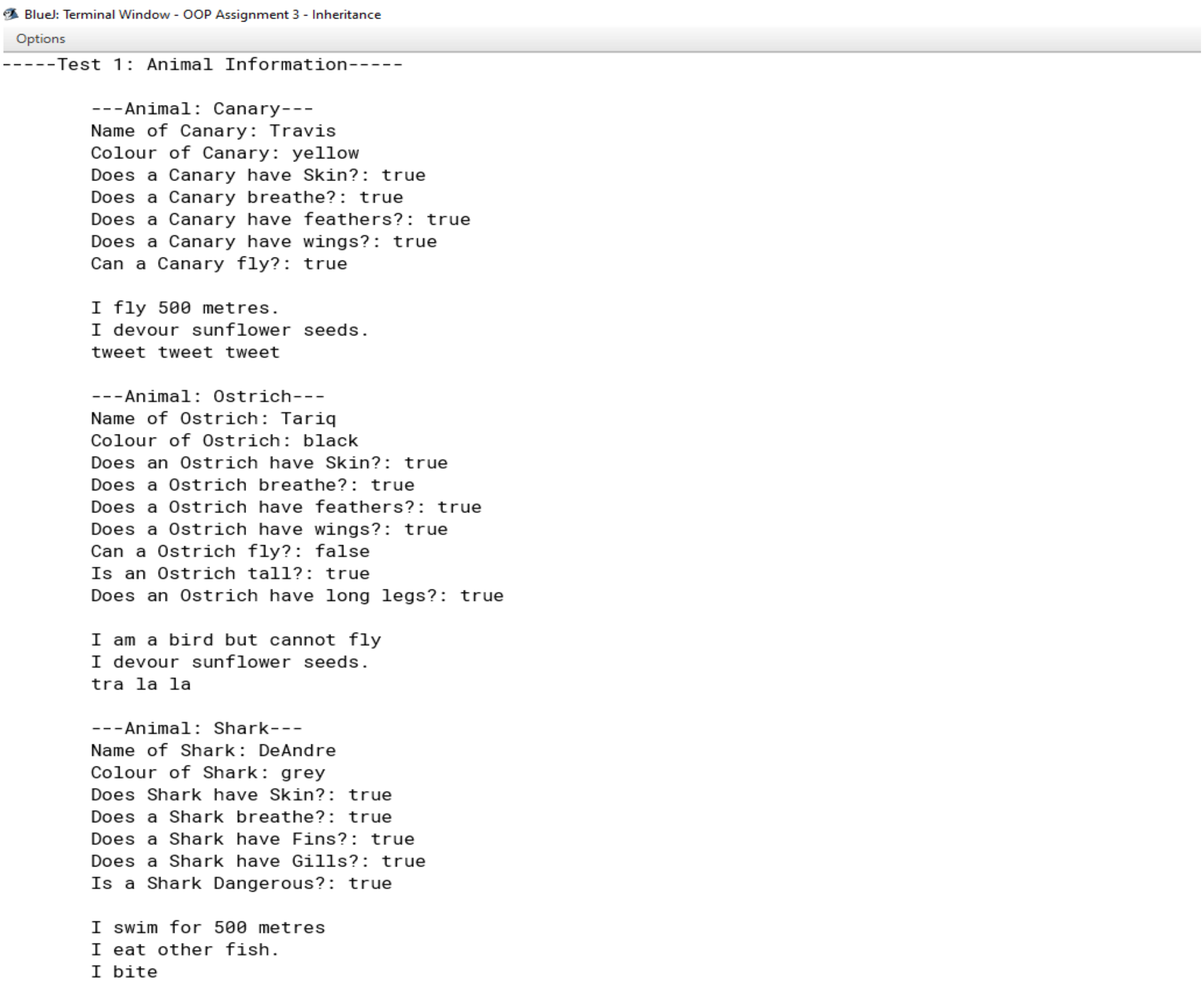
AnimalTest Code:

```
public class AnimalTest {

  public static void main(String[] args) {

    //The two tests are called.

    animalInfo();

    animalEquality();

  }


  private static void animalInfo() {


    System.out.println("-----Test 1: Animal Information-----");


    //An array of animals of size 4 is declared

    Animal[] animals = new Animal[4];
```

```
//Each array element and corresponding names are declared
      animals[0] = new Canary("Travis");
      animals[1] = new Ostrich("Tariq");
      animals[2] = new Shark("DeAndre");
      animals[3] = new Trout("Bishop");

      //A for loop is used to parse each animal in the array as it loops
      for (Animal animal : animals) {
         System.out.println();
         //The animals corresponding toString() method is called
         System.out.print(animal);
         //Also each animal is called to move and eat.
         animal.move(500);
         animal.eat();
         //The sing method will also be called if the object in the array is a bird
         if (animal instanceof Bird ){
            ((Bird) animal).sing();
         }
         //Or if it's a shark, it will be called upon to bite.
         else if (animal instanceof Shark ){
            ((Shark) animal).bite();
         }
      }
   }

   private static void animalEquality(){
      System.out.println("\n----Test 2: Animal Equality----");
      //Animal array with 10 animals.
      Animal[] animals = new Animal[10];
```

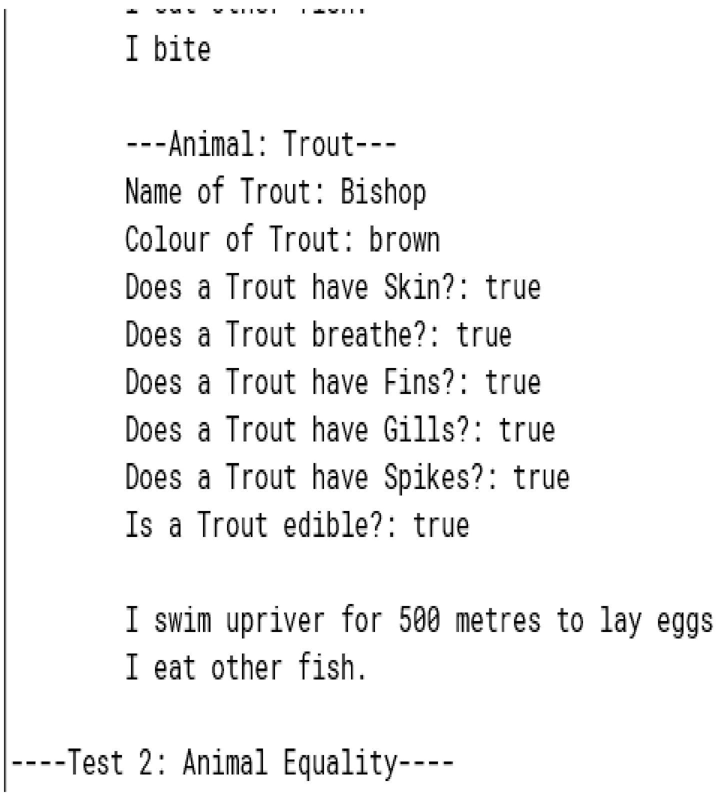
```
  //Canaries
      animals[0] = new Canary("Tom");
      animals[1] = new Canary("Tom");
      //Ostriches
      animals[2] = new Ostrich("Red");
      animals[3] = new Ostrich("John");
      //Trout
      animals[4] = new Trout("Lee");
      animals[5] = new Trout("Greg");
      animals[6] = new Trout("Lee");
      //Sharks
      animals[7] = new Shark("Joe");
      animals[8] = new Shark("Joe");
      animals[9] = new Shark("Will");

      //Running Equality test between different animals
      //If animals are of the same type (AND SAME NAME) they are considered equal.
      System.out.println("\n\tCanary equals Canary (Same Name): "+animals[0].equals(animals[1]));
      System.out.println("\n\tCanary equals Shark: "+animals[0].equals(animals[7]));
      System.out.println("\n\tOstrich equals Trout: "+animals[2].equals(animals[4]));
      System.out.println("\n\tTrout equals Trout (Different Name): "+animals[4].equals(animals[5]));
      System.out.println("\n\tTrout equals Trout (Same Name): "+animals[4].equals(animals[6]));
      System.out.println("\n\tShark equals Ostrich: "+animals[7].equals(animals[3]));
      System.out.println("\n\tTrout equals Canary: "+animals[5].equals(animals[1]));
      System.out.println("\n\tOstrich equals Ostrich (Different Name): "+animals[2].equals(animals[3]));
      System.out.println("\n\tShark equals Shark (Same Name): "+animals[7].equals(animals[8]));
      System.out.println("\n\tShark equals Shark (Different Name): "+animals[7].equals(animals[9]));
   }
}
```

AnimalTest Test 1 Output:


BlueJ: Terminal Window - OOP Assignment 3 - Inheritance

Options

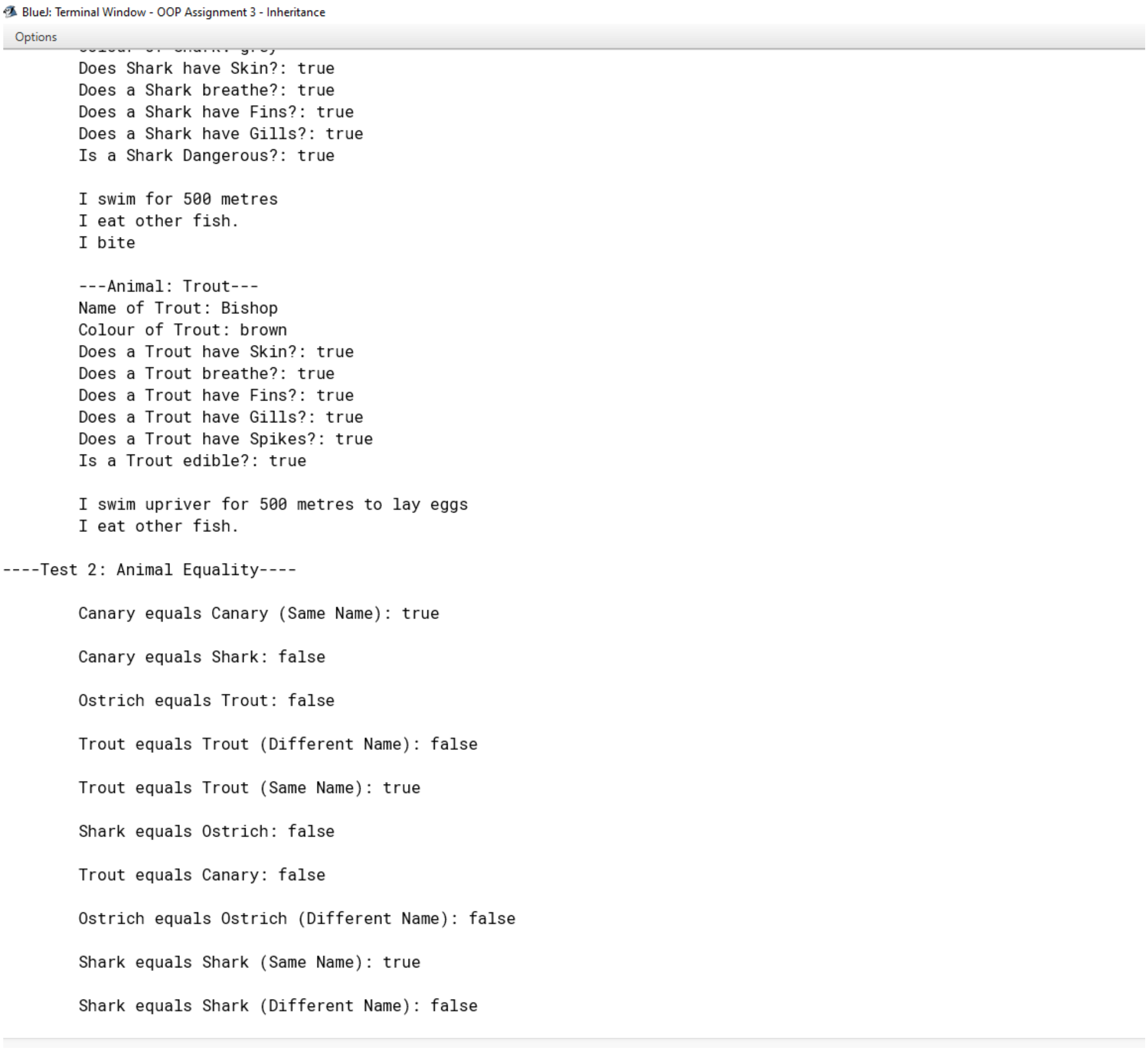
```
-----Test 1: Animal Information-----

        ---Animal: Canary---
        Name of Canary: Travis
        Colour of Canary: yellow
        Does a Canary have Skin?: true
        Does a Canary breathe?: true
        Does a Canary have feathers?: true
        Does a Canary have wings?: true
        Can a Canary fly?: true

        I fly 500 metres.
        I devour sunflower seeds.
        tweet tweet tweet

        ---Animal: Ostrich---
        Name of Ostrich: Tariq
        Colour of Ostrich: black
        Does an Ostrich have Skin?: true
        Does a Ostrich breathe?: true
        Does a Ostrich have feathers?: true
        Does a Ostrich have wings?: true
        Can a Ostrich fly?: false
        Is an Ostrich tall?: true
        Does an Ostrich have long legs?: true

        I am a bird but cannot fly
        I devour sunflower seeds.
        tra la la

        ---Animal: Shark---
        Name of Shark: DeAndre
        Colour of Shark: grey
        Does Shark have Skin?: true
        Does a Shark breathe?: true
        Does a Shark have Fins?: true
        Does a Shark have Gills?: true
        Is a Shark Dangerous?: true

        I swim for 500 metres
        I eat other fish.
        I bite
```



```
        I bite
```

```
        ---Animal: Trout---
        Name of Trout: Bishop
        Colour of Trout: brown
        Does a Trout have Skin?: true
        Does a Trout breathe?: true
        Does a Trout have Fins?: true
        Does a Trout have Gills?: true
        Does a Trout have Spikes?: true
        Is a Trout edible?: true

        I swim upriver for 500 metres to lay eggs
        I eat other fish.

----Test 2: Animal Equality----
```

AnimalTest Test 2 Output:



```
        Does Shark have Skin?: true
        Does a Shark breathe?: true
        Does a Shark have Fins?: true
        Does a Shark have Gills?: true
        Is a Shark Dangerous?: true

        I swim for 500 metres
        I eat other fish.
        I bite

        ---Animal: Trout---
        Name of Trout: Bishop
        Colour of Trout: brown
        Does a Trout have Skin?: true
        Does a Trout breathe?: true
        Does a Trout have Fins?: true
        Does a Trout have Gills?: true
        Does a Trout have Spikes?: true
        Is a Trout edible?: true

        I swim upriver for 500 metres to lay eggs
        I eat other fish.

----Test 2: Animal Equality----

        Canary equals Canary (Same Name): true

        Canary equals Shark: false

        Ostrich equals Trout: false

        Trout equals Trout (Different Name): false

        Trout equals Trout (Same Name): true

        Shark equals Ostrich: false

        Trout equals Canary: false

        Ostrich equals Ostrich (Different Name): false

        Shark equals Shark (Same Name): true

        Shark equals Shark (Different Name): false
```

Canary Code:

```
public class Canary extends Bird
{
    //Name associated with this specific Canary.
    String name;

    /**
     * Constructor for objects of class Canary
     */
    public Canary(String name)
    {
        //Constructor of superclass Bird is called
        super();
        this.name = name;
        //Colour inherited from the superclass is overwritten
        colour = "yellow";
    }

    /**
     *Methods
     */
    //Sing method is overridden from its superclass
    @Override
    public void sing(){
        System.out.println("\ttweet tweet tweet");
    }

    /**
     *toString and equals methods.
     */
```

```
//The equals method needs to be overridden because all classes inherit from the default Object class
@Override
public boolean equals(Object o) {
  //Object is checked to see if it's an instance of Canary

  if (!(o instanceof Canary )) {
    return false;
  }

   //If it is, object passed into method is cast to an object of type canary
  //and its variables are also compared
  Canary p = (Canary)o;

  //Each variable of a typical canary is compared with the object passed into the method
  //If there is a discrepancy, the method outputs false

  if (hasFeathers != p.hasFeathers){
    return false;
  }
  //Compares two strings using the Java String class equals method that is not overridden
  //If two strings are equal, equals() returns true
  else if (!(name.equals(p.name))){
    return false;
  }
  else if (hasWings != p.hasWings){
    return false;
  }
```

```
      else if(hasSkin != p.hasSkin){
          return false;
      }
      //If all checks passed, and the final check is true, that is returned to main.
      //If the final check is false, false is returned to main.
      else return breathes == p.breathes;
  }


  //Every class also has a default implementation of a toString method, therefore it needs to be
overridden
  @Override
  public String toString(){
      return
              "\t---Animal: Canary---"+
              "\n\tName of Canary: "+name+
      //Canary's superclass, bird will use the get method that it has inherited from its superclass animal.
              "\n\tColour of Canary: "+super.getColour()+
              "\n\tDoes a Canary have Skin?: "+super.hasSkin()+
              "\n\tDoes a Canary breathe?: "+super.getBreathe()+
              "\n\tDoes a Canary have feathers?: "+hasFeathers()+
              "\n\tDoes a Canary have wings?: "+hasWings()+
              "\n\tCan a Canary fly?: "+getFly()+
              "\n";
  }
}
```

Ostrich Code:

```
public class Ostrich extends Bird {
   //Name of Ostrich
   String name;
   boolean isTall;
   boolean hasLongLegs;

   /**
    * Constructor for objects of class Ostrich
    */

   public Ostrich(String name)
   {
      //Constructor of superclass Bird is called
      super();
      this.name = name;
      isTall = true;
      hasLongLegs = true;
      flies = false;
   }

   /**
    *toString and equals methods.
    */

   //The equals method needs to be overridden because all classes inherit from the default Object class
   @Override
   public boolean equals(Object o) {
      if (!(o instanceof Ostrich )) {
         return false;
      }
      //p is cast to Ostrich.
      Ostrich p = (Ostrich)o;


      if (isTall != p.isTall){
         return false;
      }
      else if (hasLongLegs != p.hasLongLegs){
         return false;
      }
      //true is returned if there's a match
      else if (!(name.equals(p.name))){
         return false;
      }


      else if (flies != p.flies){
         return false;
      }


      else if(hasSkin != p.hasSkin){
         return false;
      }
      else return breathes == p.breathes;
   }
```

```
//Every class also has a default implementation of a toString method, therefore it needs to be overridden
 @Override
 public String toString(){
   return
       "\t---Animal: Ostrich---"+
       "\n\tName of Ostrich: "+name+
       //Ostrich's superclass, bird will use the get method that it has inherited from its superclass animal.
       "\n\tColour of Ostrich: "+super.getColour()+
       "\n\tDoes an Ostrich have Skin?: "+super.hasSkin()+
       "\n\tDoes a Ostrich breathe?: "+super.getBreathe()+
       "\n\tDoes a Ostrich have feathers?: "+hasFeathers()+
       "\n\tDoes a Ostrich have wings?: "+hasWings()+
       "\n\tCan a Ostrich fly?: "+getFly()+
       "\n\tIs an Ostrich tall?: "+isTall+
       "\n\tDoes an Ostrich have long legs?: "+hasLongLegs+
       "\n";
 }
}
```

Fish Code:

```
public abstract class Fish extends Animal {
    //Values and properties associated with all fish
    boolean hasFins;
    boolean hasGills;

    public Fish()
    {
        //Constructor of superclass Animal is called
        super();
        hasFins = true;
        hasGills = true;
    }

    //Fish methods that each individual type of fish will override
    @Override
    public void eat() {
        System.out.println("\n\tI eat other fish.");
    }

    @Override
    public void move(int distance) {
        System.out.printf("\n\tI swim for %d metres", distance);
    }
    public boolean getFins() {
        return hasFins;
    }

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

Shark Code:

```
public class Shark extends Fish {
   //The name of the shark
   String name;

   //Properties associated with all sharks
   boolean isDangerous;

   /**
    * Constructor for objects of class Shark
    */
   public Shark(String name) {
      //Constructor of superclass Fish is called
      super();
      this.name = name;
      isDangerous = true;
   }

   /**
    * Methods
    */
   //Bite will be implemented as a method as it is an action.
   public void bite() {
      System.out.println("\tI bite");
   }



   /**
    *toString and equals methods.
    */

   //The equals method needs to be overridden because all classes inherit from the default Object class
```

```
@Override
public boolean equals(Object o) {

    if (!(o instanceof Shark)) {
        return false;
    }

    //p is cast to shark if it is an instance of shark
    Shark p = (Shark)o;

    if (isDangerous !=p.isDangerous){
        return false;
    }

    //Name of shark is compared, true is returned if there's a match
     else if (!(name.equals(p.name))){
         return false;
     }

    else if(hasFins != p.hasFins){
        return false;
    }
    else if(hasGills != p.hasGills) {
        return false;
    }

    else if(hasSkin != p.hasSkin){
        return false;
    }
    else return breathes == p.breathes;
}
```

```
    //Default toString() method is overridden

    @Override

    public String toString() {

        return

                "\t---Animal: Shark---" +

                "\n\tName of Shark: " + name +

                //Shark's superclass, shark will use the get method that it has inherited from its superclass animal.

                "\n\tColour of Shark: " + super.getColour() +

                "\n\tDoes Shark have Skin?: " + super.hasSkin() +

                "\n\tDoes a Shark breathe?: " + super.getBreathe() +

                "\n\tDoes a Shark have Fins?: " + getFins() +

                "\n\tDoes a Shark have Gills?: " + getGills() +

                "\n\tIs a Shark Dangerous?: " + isDangerous+

                "\n";

    }

}
```

Trout Code:

```
public class Trout extends Fish {
  //Name of Trout
  String name;

  //Properties and values associated with each fish
  boolean hasSpikes;
  boolean isEdible;

  /**
   * Constructor for objects of class Trout
   */
  public Trout(String name) {
    //Constructor of superclass Fish is called
    super();
    this.name = name;
    hasSpikes = true;
    isEdible = true;
    //Colour of this particular fish is set
    colour = "brown";
  }

  /**
   * Methods
   */
  //The move method in superclass fish is overridden to print a message specific to the trout
  @Override
  public void move(int distance){
    System.out.printf("\n\tI swim upriver for %d metres to lay eggs", distance);
  }
```

```
/**
 *toString and equals methods.
 */
//The equals method needs to be overridden because all classes inherit from the default Object class
@Override
public boolean equals(Object o) {
    if (!(o instanceof Trout)) {
        return false;
    }

        Trout p = (Trout)o;

    if (hasSpikes != p.hasSpikes){
        return false;
    }
    //true is returned if there's a match
    else if (!(name.equals(p.name))){
        return false;
    }

    else if(hasFins != p.hasFins){
        return false;
    }
    else if(hasGills != p.hasGills) {
        return false;
    }

    else if(hasSkin != p.hasSkin){
        return false;
    }
    else return breathes == p.breathes;
}
```

```
 //Default toString() method is overridden
 @Override
 public String toString() {
    return
          "\t---Animal: Trout---" +
          "\n\tName of Trout: " + name +
          //Trout's superclass, shark will use the get method that it has inherited from its superclass animal.
          "\n\tColour of Trout: " + super.getColour() +
          "\n\tDoes a Trout have Skin?: " + super.hasSkin() +
          "\n\tDoes a Trout breathe?: " + super.getBreathe() +
          "\n\tDoes a Trout have Fins?: " + getFins() +
          "\n\tDoes a Trout have Gills?: " + getGills() +
          "\n\tDoes a Trout have Spikes?: " + hasSpikes +
          "\n\tIs a Trout edible?: " + isEdible +
          "\n";
 }

}
```

Explanation of Code:

The move method in the bird class includes an if statement. This if statement checks whether the subclass that has inherited the “flies” variable from its superclass, changed the variable or not. If a subclass changed the variable the false i.e., it can’t fly, the method will ensure a different move statement is printed to the console. This method in the bird class overrides its superclass (animal) move method. All animals move differently, therefore it must be overridden. An eat method is implemented in the same way.

Not all animals sing however, so a sing method is declared in the bird class. Each leaf class i.e., (Ostrich) of bird will have to override this default sing method if it sings differently.

All classes in Java have implementations of toString and equals methods. Therefore, they need to be overridden. The equals method checks to see if the object passed into the method is equal to the class it was called from. It checks to see if the object passed is an instance of the animal it is comparing against. If it is, the object passed is casted to that specific animal type, and has all variables associated with that animal checked against. For two objects to be equal, they need to have the same name, the default implementation of equals is used to compare two strings i.e., the names of two objects being compared (outputs true if two strings are the same). If it passes all checks and is equal to the object is being checked against, the method returns true to main.

In a similar fashion, the toString method also needs to be overridden. Each leaf class has a toString method, which returns a string of the specific characteristics of the animal in question.

Animal, bird and fish classes are declared as abstract as they cannot be instantiated.

In the Shark class, bite was implemented as a method, as biting is an action, just like move or sing.

Inside the testAnimal class, animals are stored in arrays. Inside the test1 method, each animal in the array is looped through and has its state printed to the screen using the toString method. Each animal is also called to move 500. Different animals will move differently and how they move is printed to the screen. Each animal is also called to eat. Not all animals can sing, therefore the animal selected on that iteration of the loop is checked to see if it an instance of Bird. If it is, it is called to sing. A shark is the only animal that can bite in the program. An else if statement is used to check if the animal is a shark, if it is, it is called to bite.