CT2106 Assignment 3

Michael Mc Curtin

ID: 21459584

Project Description

The project represents a hierarchy of the **Animal** Kingdom.

Animals are either categorised as **Birds** or **Fish** based on physical characteristics (e.g. whether they have feathers or gills) and based on movement (whether they can swim or can fly).

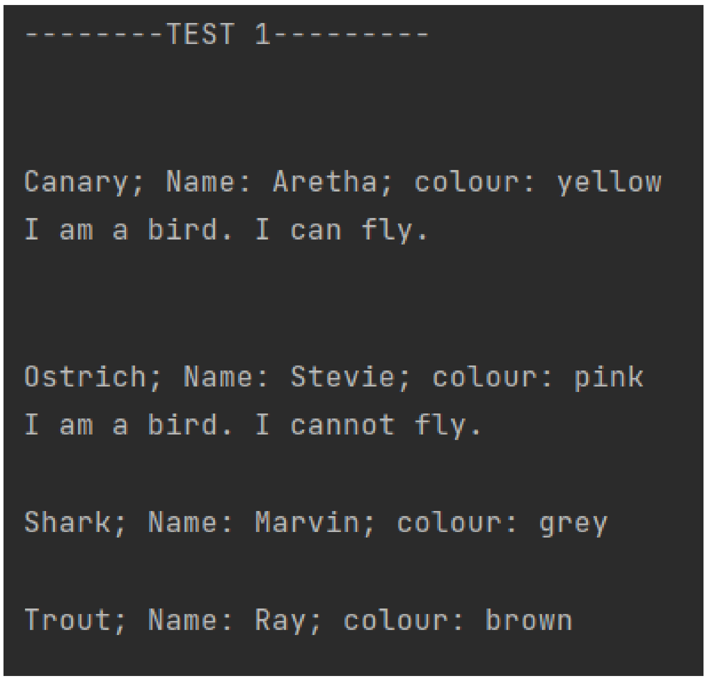
The bird species **Canary** and **Ostrich**, and the fish species **Shark** and **Trout** further descend from this categorisation.

The **AnimalTest** class creates several named animals of each type. It then goes on to run comparisons between them to see if they are logically equivalent to one another based on their characteristics.

AnimalTest class

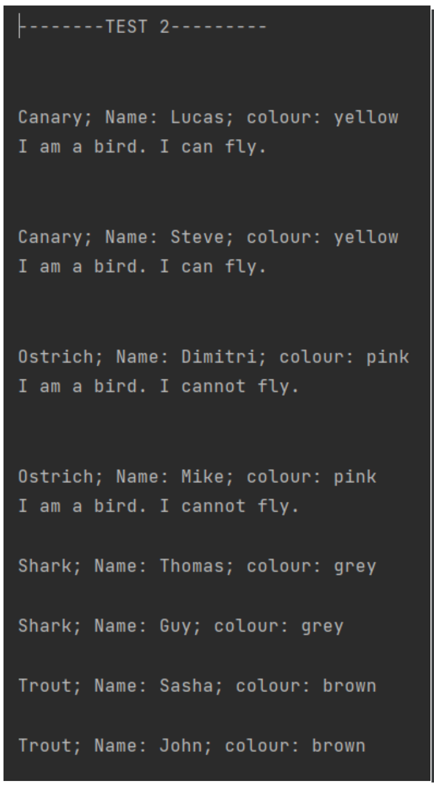
```
public class AnimalTest {
    public static void main(String[] args) {
        test1();
        test2();
    }
    public static void test1() {

        System.out.println("--------TEST 1---------\n");

        // create array of 4 animals of the 4 different types

        Animal[] animals = new Animal[4];

        animals[0] = new Canary("Aretha");
        animals[1] = new Ostrich("Stevie");
        animals[2] = new Shark("Marvin");
        animals[3] = new Trout("Ray");

        // print out animal toString information

        for (int i = 0; i < animals.length; i++) {
            System.out.println(animals[i]);
        }
    }
    public static void test2() {

        System.out.println("--------TEST 2---------\n");

        // create array of 8 animals of the 4 different types


        Animal[] animals = new Animal[8];


        animals[0] = new Canary("Lucas");
        animals[1] = new Canary("Steve");
        animals[2] = new Ostrich("Dimitri");
        animals[3] = new Ostrich("Mike");
        animals[4] = new Shark("Thomas");
        animals[5] = new Shark("Guy");
        animals[6] = new Trout("Sasha");
        animals[7] = new Trout("John");


        // print out animal toString information


        for (int i = 0; i < animals.length; i++) {
            System.out.println(animals[i]);
        }


        // print out whether each animal is equal to the next animal

        for (int i = 0, j = 1; j < animals.length; i++) {
            System.out.println(String.format("\nAnimal at index %d equals
Animal at index %d ?\n", i, j));
            System.out.println(animals[i].equals(animals[j]));
            j++;
        }

    }
}
```

AnimalTest test1 Output

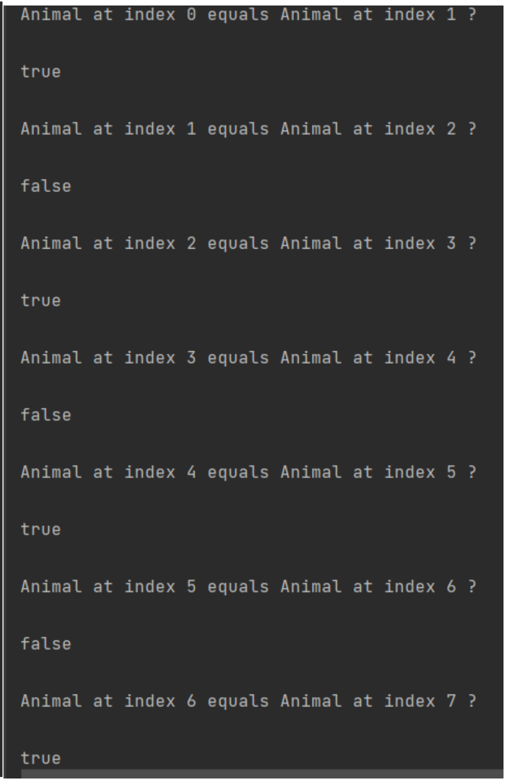


```
--------TEST 1---------


Canary; Name: Aretha; colour: yellow
I am a bird. I can fly.


Ostrich; Name: Stevie; colour: pink
I am a bird. I cannot fly.

Shark; Name: Marvin; colour: grey

Trout; Name: Ray; colour: brown
```

AnimalTest test2 Output



```
--------TEST 2---------


Canary; Name: Lucas; colour: yellow
I am a bird. I can fly.


Canary; Name: Steve; colour: yellow
I am a bird. I can fly.


Ostrich; Name: Dimitri; colour: pink
I am a bird. I cannot fly.


Ostrich; Name: Mike; colour: pink
I am a bird. I cannot fly.

Shark; Name: Thomas; colour: grey

Shark; Name: Guy; colour: grey

Trout; Name: Sasha; colour: brown

Trout; Name: John; colour: brown
```



```
Animal at index 0 equals Animal at index 1 ?

true

Animal at index 1 equals Animal at index 2 ?

false

Animal at index 2 equals Animal at index 3 ?

true

Animal at index 3 equals Animal at index 4 ?

false

Animal at index 4 equals Animal at index 5 ?

true

Animal at index 5 equals Animal at index 6 ?

false

Animal at index 6 equals Animal at index 7 ?

true
```

Bird class

```
public abstract class Bird extends Animal
{
    //instance variables (fields)
    boolean hasFeathers;
    boolean hasWings;
    boolean flies;

    /**
     * Constructor for objects of class Bird
     */
    public Bird()
    {
        super(); //calls the constructor of its superclass  - Animal
        colour = "black"; //overrides the value of colour inherited from
Animal
        hasFeathers = true; //all the subclasses of Fish inherit this
property and value
        hasWings = true; //all the subclasses of Fish inherit this property
and value
        flies = true; //all the subclasses of Fish inherit this property
and value
    }

    /**
     * move method overrides the move method
     * inherited from superclass Animal
     */

    @Override
    public void move(int distance){

        // not all birds fly so change movement message based on this

        if (flies = true) {
            System.out.printf("I fly %d metres \n", distance);
        }
        else {
            System.out.printf("I run %d metres \n", distance);
        }

    }
    /**
     * toString method overrides the toString method
     * inherited from superclass Animal
     */
    @Override
    public String toString(){

        // say whether the bird can fly or not

        String condition;

        if (this.flies){
            condition = "can";
        }

        else {
            condition = "cannot";
        }
        String strng = String.format("I am a bird. I %s fly.\n",
condition);
        return strng;
    }


    /**
     * equals method overrides the equals method
     * inherited from superclass Animal
     */

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;  // check if object is this object
        if (!(o instanceof Bird bird)) return false; // check if object is
a Bird
        if (!super.equals(o)) return false; // check if object is an Animal
        return hasFeathers == bird.hasFeathers && hasWings == bird.hasWings
&& flies == bird.flies;
    }

    /**
     * sing method
     * All subclasses inherit this method
     */

    public void sing(){
        System.out.println("tra la la");
    }

    /**
     * 'getter' method for the hasWings field
     *  All subclasses inherit this method
     */
    public boolean hasWings(){
        return hasWings;
    }

    /**
     * 'getter' method for the hasFeathers field
     *  All subclasses inherit this method
     */
```

```java
    public boolean hasFeathers(){
        return hasFeathers;
    }
}
```

Canary class

Ostrich class

```
public class Ostrich extends Bird
{

    String name; // the name of this Ostrich
    String legs; // leg information

    String height;

    /**
     * Constructor for objects of class Ostrich
     */
    public Ostrich(String name)
    {
        super(); // call the constructor of the superclass Bird
        this.name = name;
        colour = "pink"; // this overrides the value inherited from Bird
        flies = false;
        legs = "thin and long";
        height = "tall";
    }

    /**
     * toString method overrides the toString method
     * inherited from superclass Bird
     */

    @Override
    public String toString(){
        String strng;

        strng = String.format("\nOstrich; Name: %s; colour: %s\n", name,
colour);
        strng += super.toString();
        return strng;
    }

    /**
     * equals method defines how equality is defined between
     * the instances of the Ostrich class
     * param Object
     * return true or false depending on whether the input object is
     * equal to this Ostrich object
     */

    @Override
    public boolean equals(Object o) {
        if (this == o) return true; // check if object is this object
        if (!(o instanceof Ostrich ostrich)) return false; // check if
object is an Ostrich
        if (!super.equals(o)) return false; // check if object is a Bird
        return legs.equals(ostrich.legs) && height.equals(ostrich.height);
    }

}
```

Fish class

```java
import java.util.Objects;

public abstract class Fish extends Animal
{
    //instance variables (fields)
    boolean hasFins;
    boolean hasGills;
    boolean swims;

    boolean dangerous;
    boolean edible;

    /**
     * Constructor for objects of class Fish
     */
    public Fish()
    {
        super(); //calls the constructor of its superclass  - Animal
        colour = "blue"; //sets the value of colour inherited from Animal
        hasFins = true; //all the subclasses of Fish inherit this property
and value
        hasGills = true; //all the subclasses of Fish inherit this property
and value
        swims = true; //all the subclasses of Fish inherit this property
and value
        edible = false;
        dangerous = false;
    }

    /**
     * move method overrides the move method
     * inherited from superclass Animal
     */
    @Override // good programming practice to use @Override to denote
overridden methods
    public void move(int distance){
            System.out.printf("I swim %d metres \n", distance);
    }

    /**
     * bite method that all fish have
     */
    public void bite(){

        // fish will only cause harm with its bite if it is dangerous

        if (dangerous) {
            System.out.println("nomnomnom");
        }

        else {
            System.out.println("slurp slurp slurp");

        }
    }
    /**
     * equals method defines how equality is defined between
     * the instances of the Fish class
     * param Object
     * return true or false depending on whether the input object is
     * equal to this Fish object
     */

    @Override
    public boolean equals(Object o) {
        if (this == o) return true; // check if object is this object
        if (!(o instanceof Fish fish)) return false; // check if object is
a Fish
        if (!super.equals(o)) return false; // check if object is an Animal
        return hasFins == fish.hasFins && hasGills == fish.hasGills &&
swims == fish.swims;
    }

    /**
     * 'getter' method for the hasGills field
     */
    public boolean hasGills(){
        return hasGills;
    }

    /**
     * 'getter' method for the hasFins field
     */
    public boolean hasFins(){
        return hasFins;
    }
}
```

Shark class

```java
import java.util.Objects;

public class Shark extends Fish
{

    String name; // the name of this Shark
    Boolean dangerous;

    /**
     * Constructor for objects of class Ostrich
     */
    public Shark(String name)
    {
        super(); // call the constructor of the superclass Bird
        this.name = name;
        dangerous = true;
        colour = "grey";
    }

    /**
     * bite method overrides the bite method
     * inherited from superclass Fish
     */

    @Override
    public void bite(){
        System.out.println("Shark bite! -90 HP");
    }

    /**
     * toString method overrides the toString method
     * inherited from superclass Fish
     */
    @Override
    public String toString(){
        String strng = String.format("Shark; Name: %s; colour: %s\n", name,
colour);
        return strng;
    }
    /**
     * equals method defines how equality is defined between
     * the instances of the Shark class
     * param Object
     * return true or false depending on whether the input object is
     * equal to this Shark object
     */

    @Override
    public boolean equals(Object o) {
        if (this == o) return true; // check if object is this object
        if (!(o instanceof Shark shark)) return false; // check if object
is a Shark
        if (!super.equals(o)) return false; // check if object is a Fish
        return dangerous.equals(shark.dangerous) &&
getColour().equals(shark.getColour());
    }

}
```

Trout class

```
public class Trout extends Fish
{

    String name; // the name of this Shark
    Boolean hasSpikes;
    Boolean swimsUpriver;

    /**
     * Constructor for objects of class Ostrich
     */
    public Trout(String name) {
        super(); // call the constructor of the superclass Bird
        this.name = name;
        this.colour = "brown";
        edible = true;
        hasSpikes = true;
    }

    /**
     * toString method overrides the toString method
     * inherited from superclass Fish
     */

    @Override
    public String toString(){
        String strng = String.format("Trout; Name: %s; colour: %s\n", name,
colour);
        return strng;
    }

    @Override // good programming practice to use @Override to denote
overridden methods
    public void move(int distance){

        System.out.println(String.format("I swim %d metres upriver to lay
eggs.\n", distance));
    }


    /**
     * equals method defines how equality is defined between
     * the instances of the Trout class
     * param Object
     * return true or false depending on whether the input object is
     * equal to this Trout object
     */

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;  // check if object is this object
        if (!(o instanceof Trout trout)) return false; // check if object
is a Trout
        if (!super.equals(o)) return false; // check if object is a Fish
        return hasSpikes.equals(trout.hasSpikes) &&  edible == trout.edible
&& getColour().equals(trout.getColour());
    }

}
```

Code Explanation

I found that the easiest way to implement condition 1(c) was to make the ToString method of the Bird class to explain whether the bird could fly or not. I then called Bird’s ToString method in each sub-class’s ToString method and concatenated the strings.

Ostrich’s ToString method will therefore become

> *Ostrich; Name:* [name]*; colour: pink*
>
> *I am a bird. I cannot fly.*