Assignment 3 - CT2106 Object Oriented Programming

Daniel Hannon (19484286)

November 2020

1 Description

With this Assignment, we had to use class inheritance to create four different animal types, each with several unique features. As they were all child classes of the parent class **Animal** they could all be stored in an array as they all had some common traits.

2 Code

2.1 AnimalTest

```
public class Animaltester {
  public Animaltester() {
    //Constructor that does nothing
  }

  public static void main(String args[]) {
    Animaltester tests = new Animaltester();
    System.out.println("Running test 1");
    tests.test1();
    System.out.println("\nRunning test 2");
    tests.test2();
  }

  public void test1() {
    Animal[] animals = new Animal[4];
    animals[0] = new Canary("Jeff");
    animals[1] = new Ostritch("Paul");
    animals[2] = new Shark("Crikey");
    animals[3] = new Trout("Steve");
    //Loops through and prints everything toString() method call is redundant
    for(int i =0; i < animals.length; i++) {
      System.out.print(animals[i]);
    }

  }

  public void test2() {
    Animal[] animals = new Animal[4];
    animals[0] = new Canary("Jeff");
    animals[1] = new Ostritch("Paul");
    animals[2] = new Shark("Crikey");
    animals[3] = new Trout("Steve");
    System.out.println("Checking for equivalent animals");
    for(int i = 0; i < animals.length; i++) {
      System.out.print("Animal " + i +" ");
      for (int j = 0; j < animals.length; j++) {
        //Prints the values of the equals methods()
        System.out.print(" " + animals[i].equals(animals[j]));
      }
      System.out.print("\n");
    }
    /*Proof of ostritch*/
    Ostritch jeff = new Ostritch("Jeff");
    jeff.move(10);
```

```
  }
}
```

2.1.1 AnimalTest outputs

```
[daniel@Void3 Assignment 3]$ java Animaltester
Running test 1
Canary; name: Jeff; colour: yellow has Feathers; true; Has wings: true; Flies: true; breathes: true; has skin: true
Ostritch; Name: Paul; Color: black; Is Tall: true; Has Long Thin Legs: true; Flies: false; Has Feathers: true; has wings: tru
e; Breathes: true; Has Skin: true
Shark Name: Crikey; Color: grey; Can Bite: true; Is Dangerous: true; Has Gills: true; Has Fins: true; Can Swim: true; Breathe
s: true
Trout Name: Steve; Color: Brown; Has Spikes: true; Is Edible: true; Swims Upstream to lay eggs: true; Has Gills: true; Has Fi
ns: true; Can Swim: true; Breathes: true

Running test 2
Checking for equivalent animals
Animal 0  true false false false
Animal 1  false true false false
Animal 2  false false true false
Animal 3  false false false true
I am a bird, but cannot fly
I run 10 metres
[daniel@Void3 Assignment 3]$
```

2.2 Canary

```
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;
    this.colour = "yellow"; // this overrides the value inherited from Bird
  }
  public String getName() {
    return this.name;
  }
  /**
   * 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(){
    String strng ="";
    strng+= "Canary; ";
    strng+= "name: ";
```

```
    strng+= this.name;
    strng+= "; ";
    strng+= "colour: ";
    strng+= this.colour;
    strng+= " has Feathers; ";
    strng+= this.hasFeathers;
    strng+= "; Has wings: ";
    strng+= this.hasWings;
    strng+= "; Flies: ";
    strng+= this.flies;
    strng+= "; breathes: ";
    strng+= this.breathes;
    strng+= "; has skin: ";
    strng+= this.hasSkin;
    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(Object obj){
    try {
      obj = (Canary)obj; /*Gets rid of passing a string in*/
      return obj.toString().equals(this.toString());
       → /*Converts to strings and compares both strings*/
    } catch(Exception e) {
      return false;
    }
  }
}
```

2.3 Ostritch

```
public class Ostritch extends Bird{
  boolean isTall;
  boolean hasLongThinLegs;
  String name;

  Ostritch(String name) {
    super();
    this.name = name;
    this.isTall = true;
    this.hasLongThinLegs = true;
    this.flies = false;
  }

  @Override
  public String toString() {
    String output = "";
```

```
    output += "Ostritch; ";
    output += "Name: ";
    output += this.name;
    output += "; Color: ";
    output += this.colour;
    output += "; Is Tall: ";
    output += this.isTall;
    output += "; Has Long Thin Legs: ";
    output += this.hasLongThinLegs;
    output += "; Flies: ";
    output += this.flies;
    output += "; Has Feathers: ";
    output += this.hasFeathers;
    output += "; has wings: ";
    output += this.hasWings;
    output += "; Breathes: ";
    output += this.breathes;
    output += "; Has Skin: ";
    output += this.hasSkin;
    output += "\n";

    return output;
  }

  @Override
  public boolean equals(Object obj) {
    try {
      obj = (Ostritch)obj;
      return this.toString().equals(obj.toString());
    } catch(Exception e) {
      return false;
    }
  }

  public boolean isTall() {
    return this.isTall;
  }

  public boolean hasLongThinLegs() {
    return this.hasLongThinLegs;
  }

  public String getName() {
    return this.name;
  }
}
```

2.4 Fish

```
public class Fish extends Animal {
  boolean hasFins;
  boolean canSwim;
  boolean hasGills;
  public Fish() {
    this.hasFins = true;
    this.canSwim = true;
    this.hasGills = true;
```

```
  }

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

  public boolean hasFins() {
    return this.hasFins;
  }

  public boolean canSwim() {
    return this.canSwim;
  }

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

2.5 Shark

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

  public Shark(String name) {
    super();
    this.name = name;
    canBite = true;
    isDangerous = true;
  }

  public boolean canBite() {
    return this.canBite;
  }

  public boolean isDangerous() {
    return this.isDangerous;
  }

  public String getName() {
    return this.name;
  }

  @Override
  public String toString() {
    String output = "Shark";
    output += " Name: " + this.name;
    output += "; Color: " + this.colour;
    output += "; Can Bite: " + this.canBite;
    output += "; Is Dangerous: " + this.isDangerous;
    output += "; Has Gills: " + this.hasGills;
    output += "; Has Fins: " + this.hasFins;
    output += "; Can Swim: " + this.canSwim;
    output += "; Breathes: " + this.breathes;
```

```
    output += "\n";
    return output;
  }

  @Override
  public boolean equals(Object o) {
    //Type casted so I could check it
    /*The way this works is simple, it checks if canBite exists, and catches ClassCastException*/
    try {
      Shark shark = (Shark)o;
      if(shark.canBite()) return shark.toString().equals(this.toString());
    } catch(Exception e) {
      return false;
    }
    return false;
  }
}
```

2.6 Trout

```
public class Trout extends Fish{
  boolean hasSpikes;
  boolean isEdible;
  boolean swimsUpstreamToLayEggs;
  String name;

  public Trout(String name) {
    this.name = name;
    this.hasSpikes = true;
    this.colour = "Brown";
    this.isEdible = true;
    this.swimsUpstreamToLayEggs = true;
  }

  public boolean hasSpikes() {
    return this.hasSpikes;
  }

  public boolean isEdible() {
    return this.isEdible;
  }

  public boolean swimsUpstreamToLayEggs() {
    return this.swimsUpstreamToLayEggs;
  }

  @Override
  public String toString() {
    String output = "Trout";
    output += " Name: " + this.name;
    output += "; Color: " + this.colour;
    output += "; Has Spikes: " + this.hasSpikes;
    output += "; Is Edible: " + this.isEdible;
    output += "; Swims Upstream to lay eggs: " + this.swimsUpstreamToLayEggs;
    output += "; Has Gills: " + this.hasGills;
    output += "; Has Fins: " + this.hasFins;
    output += "; Can Swim: " + this.canSwim;
```

```
38      output += "; Breathes: " + this.breathes;
39      output += "\n";
40      return output;
41    }
42
43    @Override
44    public boolean equals(Object o) {
45      try {
46        o = (Trout)o;
47        return this.toString().equals(o.toString());
48      } catch(Exception e) {
49        return false;
50      }
51
52    }
53  }
```

3 Code Explanation

The Code is fairly straightforward, the **toString()** overwrites all return every feature of the animal in string format as specified.
The **equals(Object o)** uses the Equals property of the **String** type as if the objects are equal, their strings are equivalent. But in order to prevent passing a **String** type into **equalsObject o** they perform a sense check by attempting to typecast the **Object** passed into the method. This is wrapped in a **try/catch** and returns false if the object passed is not of the same type.
I reused this method for every Animal class as it is very versatile and it verifies that the **toString()** overwrite works for each Object class.
There is a distinct lack of setters as a most if not all of the characteristics of the animals do not change during its life, unless the animal is dead but that is out of scope.
Ostritches output ***I am a bird but I cannot fly*** by the addition of an if/else in the parent **Bird** class.