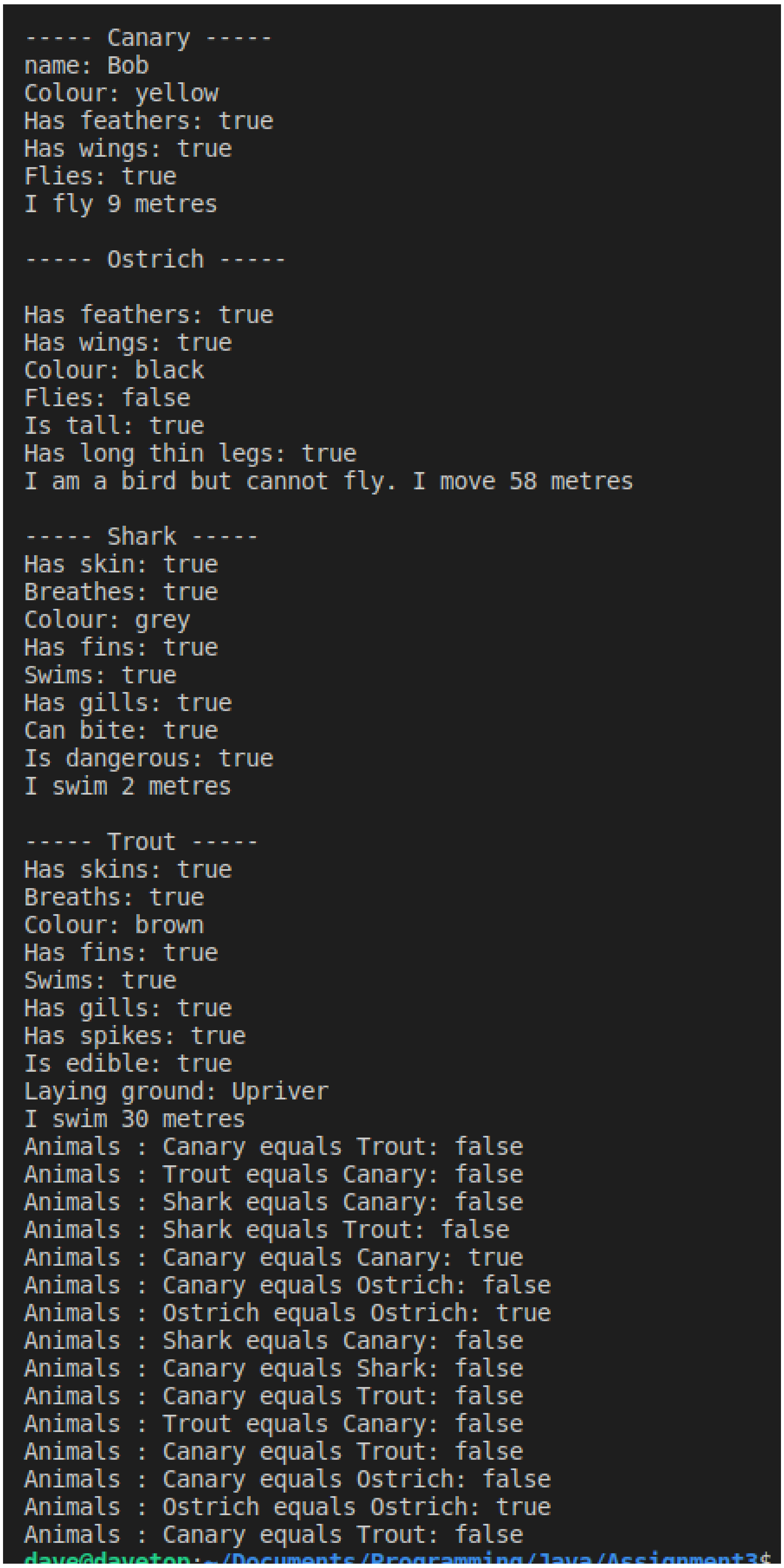
Assignment description

1) Complete the hierarchy of the animals by addomg Ostrich, Fish, Shark, Trout
2) Finish the Bird and Canary classes
3) Add overrides to the toString and equals methods for leaf classes (Canary, Ostrich, Shark, Trout)
4) Create a main method
5) Create two methods to test the toString and equals methods

```
import java.util.Random; // imports random

public class AnimalTest {

    // main method
    public static void main(String[] args) {

        // creates instance of main class
        AnimalTest test = new AnimalTest();

        // two testing methods
        test.testString();
        test.testEquals();
    }

    /*
     * This method will test the toString overrides over the classes
     * Canary, Ostrich, Shark and Trout
     */
    public void testString() {
        // Create animal array of size 4
        Animal[] animals = new Animal[4];
        Random random = new Random();
        // Populate array with animals
        animals[0] = new Canary("Bob");
        animals[1] = new Ostrich();
        animals[2] = new Shark();
        animals[3] = new Trout();

        // For loop to print out the array elements
        for (Animal animal : animals) {
            System.out.println(animal);
            animal.move(random.nextInt(100));
        }

    }

    /*
     * Method to test equals override
     */
    public void testEquals() {
        // Create animals array of size 8
        Animal[] animals = new Animal[8];
        Random random = new Random(); // create instance of random


        // populate array with animals, two of each
        animals[0] = new Canary("Bobby");
        animals[1] = new Canary("Dave");
        animals[2] = new Ostrich();
        animals[3] = new Ostrich();
        animals[4] = new Shark();
        animals[5] = new Shark();
        animals[6] = new Trout();
        animals[7] = new Trout();

        // for loop, we check 15 different times
        for (int i = 0; i < 15; i++) {
            // create two random integers of max array length
            int m = random.nextInt(animals.length - 1);
            int n = random.nextInt(animals.length - 1);

            // print out the class name that we are comparing
            // note the canary class won't always be equal to true when comparing to another canary
class
            //      this is because we have two canary instances in our array both with different names
            System.out.print("Animals : " + animals[m].getClass().getSimpleName() + " equals " +
animals[n].getClass().getSimpleName() + ": ");
            System.out.println(animals[m].equals(animals[n])); // print out their equals
        }

    }

}
```



```
----- Canary -----
name: Bob
Colour: yellow
Has feathers: true
Has wings: true
Flies: true
I fly 9 metres

----- Ostrich -----

Has feathers: true
Has wings: true
Colour: black
Flies: false
Is tall: true
Has long thin legs: true
I am a bird but cannot fly. I move 58 metres

----- Shark -----
Has skin: true
Breathes: true
Colour: grey
Has fins: true
Swims: true
Has gills: true
Can bite: true
Is dangerous: true
I swim 2 metres

----- Trout -----
Has skins: true
Breaths: true
Colour: brown
Has fins: true
Swims: true
Has gills: true
Has spikes: true
Is edible: true
Laying ground: Upriver
I swim 30 metres
Animals : Canary equals Trout: false
Animals : Trout equals Canary: false
Animals : Shark equals Canary: false
Animals : Shark equals Trout: false
Animals : Canary equals Canary: true
Animals : Canary equals Ostrich: false
Animals : Ostrich equals Ostrich: true
Animals : Shark equals Canary: false
Animals : Canary equals Shark: false
Animals : Canary equals Trout: false
Animals : Trout equals Canary: false
Animals : Canary equals Trout: false
Animals : Canary equals Ostrich: false
Animals : Ostrich equals Ostrich: true
Animals : Canary equals Trout: false
dave@davetop:~/Documents/Programming/Java/Assignment3$
```

```
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(){
        String strng ="";
        strng+= "\n----- Canary -----\n";
        strng+= "name: " + name;
        strng+= "\nColour: " + colour;
        strng+= "\nHas feathers: " + hasFeathers();
        strng+= "\nHas wings: " + hasWings();
        strng+= "\nFlies: " + flies;
        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){

        // checks if obj is equal to itself
        if(obj == this){
            return true;
        }

        // checks if the obj is a null pointer or not an instance of Canary
        if(!(obj instanceof Canary) || obj == null){
            return false;
        }

        // Casts obj to canary and checks its fields
        Canary canary = (Canary)obj;
        return canary.name.equals(this.name) &&
                canary.getColour().equals(this.getColour()) &&
                canary.hasFeathers() == this.hasFeathers() &&
                canary.hasWings() == this.hasWings() &&
                canary.flies == this.flies;
    }
}




public class Ostrich extends Bird{

    // Class fields
    boolean isTall;
    boolean hasLongThinLegs;

    // Constructor
    public Ostrich(){
        super();
        isTall = true;
        hasLongThinLegs = true;
        flies = false;
    }

    // getters
    public boolean isTall(){
        return isTall;
    }

    public boolean hasLongThinLegs(){
        return hasLongThinLegs;
    }


    // Overrides toString method
    @Override
    public String toString(){
        String strng ="";
        strng+= "\n----- Ostrich -----\n";
        strng+= "\nHas feathers: " + hasFeathers();
        strng+= "\nHas wings: " + hasWings();
        strng+= "\nColour: " + getColour();
        strng+= "\nFlies: " + flies;
        strng+="\nIs tall: " + isTall();
        strng+="\nHas long thin legs: " + hasLongThinLegs();
        return strng;
    }

    // Overrides equals method
    @Override
    public boolean equals(Object obj){
        if(obj == this)
            return true;

        if(!(obj instanceof Ostrich) || obj == null)
            return false;


        Ostrich ostrich = (Ostrich)obj;
        return ostrich.colour.equals(this.colour) &&
                ostrich.hasFeathers() == this.hasFeathers() &&
                ostrich.hasWings() == this.hasWings() &&
                ostrich.flies == this.flies &&
                ostrich.isTall() == this.isTall() &&
                ostrich.getColour().equals(this.getColour()) &&
                ostrich.hasLongThinLegs() == this.hasLongThinLegs();
    }

}
```

```
public class Fish extends Animal {
    // Class fields
    boolean hasFins;
    boolean swims;
    boolean hasGills;

    // Constructor
    public Fish(){
        super();
        hasFins = true;
        swims = true;
        hasGills = true;
    }
```

```java
    // overriding move from animal class
    // move and swim are technically the same thing
    @Override
    public void move(int distance){
        String msg = (swims ? "I swim " : "I move ") + distance + " metres";
        System.out.println(msg);
    }

    // getters
    public boolean hasFins(){
        return hasFins;
    }

    public boolean hasGills(){
        return hasGills;
    }


}
```

```java
public class Shark extends Fish{

    // Class fields
    boolean canBite;
    boolean isDangerous;

    // Constructor
    public Shark(){
        super(); // Call inherited constructor method
        canBite = true;
        isDangerous = true;
        colour = "grey";
    }


    // getters
    public boolean canBite(){
        return canBite;
    }

    public boolean isDangerous(){
        return isDangerous;
    }

    // Overrides strings method
    @Override
    public String toString(){
        String strng = "\n----- Shark -----\n";
        strng += "Has skin: " + hasSkin();
        strng +="\nBreathes: " + breathes;
        strng +="\nColour: " + getColour();
        strng += "\nHas fins: " + hasFins();
        strng += "\nSwims: " + swims;
        strng += "\nHas gills: " + hasGills();
        strng += "\nCan bite: " + canBite();
        strng += "\nIs dangerous: " + isDangerous();
        return strng;
    }

    // Overrides equals method

    @Override
    public boolean equals(Object obj){
        if(obj == this)
            return true;
        if(!(obj instanceof Shark) || obj == null)
            return false;

        Shark shark = (Shark)obj;
        return shark.hasSkin() == this.hasSkin() &&
                shark.breathes == this.breathes &&
                shark.getColour().equals(this.getColour()) &&
                shark.hasFins() == this.hasFins() &&
                shark.swims == this.swims &&
                shark.hasGills() == this.hasGills() &&
                shark.canBite() == this.canBite() &&
                shark.isDangerous() == this.isDangerous();
    }

}
```

```
public class Trout extends Fish{

    // Class fields
    boolean hasSpikes;
    boolean isEdible;
    String layingGround;

    // Constructor
    public Trout(){
        super();
        hasSpikes=true;
        isEdible=true;
        layingGround="Upriver";
        colour = "brown";
    }



    // field getters
    public boolean hasSpikes(){
        return hasSpikes;
    }
    public boolean isEdible(){
        return isEdible;
    }
    public String layingGround(){
        return layingGround;
    }

    // Overrides toString method

    @Override
    public String toString(){
        String strng = "\n----- Trout -----\n";
        strng += "Has skins: " + hasSkin();
        strng += "\nBreaths: "+breathes;
        strng+="\nColour: "+getColour();
        strng+="\nHas fins: "+hasFins();
        strng+="\nSwims: "+swims;
        strng+="\nHas gills: " + hasGills();
        strng+="\nHas spikes: " + hasSpikes();
        strng+="\nIs edible: " + isEdible();
        strng+="\nLaying ground: " + layingGround();
        return strng;
    }

    // Overrides equals method

    @Override
    public boolean equals(Object obj){
        if(obj == this)
            return true;
        if(!(obj instanceof Trout) || obj == null)
            return false;
        Trout trout = (Trout)obj;
        return trout.hasSkin() == this.hasSkin() &&
                trout.breathes == this.breathes &&
                trout.getColour() == this.getColour() &&
                trout.hasFins() == this.hasFins() &&
                trout.swims == this.swims &&
                trout.hasGills() == this.hasGills() &&
                trout.hasSpikes() == this.hasSpikes() &&
                trout.isEdible() == this.isEdible() &&
                trout.layingGround().equals(this.layingGround());
    }


}
```

-------------------------------- Brief explaination of some Code -----------------------------------

My implementation of how the Ostrich move method works.
A requirment in the assignment was to print out when the move method for the Ostrich was called that the ostrich cannot fly even though it is a bird.
I did this by using a ternary operator in the Bird method (That the ostrich extends from).
I assigned two different message depending on whether the bird (in this case Ostrich) was able to fly based on its flies boolean field. If true it would say “I fly x metres” if false it would say “I am a bird but cannot fly. I move x metres”.
This way I didn’t need to override the method or create a new one for the Ostrich. This also lets future classes that extend from bird that cannot fly to benefit from this implementation