Practice Exercises: Teaching Block 2

- Data types
- Arrays
- Compiler & Runtime errors
- Objects, Inheritance and Abstract classes
- Other practice exercise types: “Fill in the gaps” + “Predict the question”

This set of exercises is in addition to those included directly in lecture slides (and extra reading materials), which you should also attempt.

Question 1

- Are the following array declarations valid or invalid? If any are invalid, write the correct declaration(s).

```
double numbers = {3.5, 6, 2.6, 8.0};

int[] marks = int[60];

char letters[] = {a, b, c, d, e, f};

String[] books = {Java, SQL, PHP};
```

Question 2

- What is the output of this program? Explain your answer.

```
public class Test_Q2 {
  public static void main(String[] args) {
    int[] intArray = new int[5];
    for (int i=0; i<=intArray.length;i++) {
      intArray[i] = i;
    }
    System.out.println(intArray);
  }
}
```

Question 3

- Assume we have a **`Flower`** class (*), and two of its methods are **`setColour(String colour)`** and **`setHeight(double height)`**. What is wrong with the following code? Explain your answer.

```
Flower[] f = new Flower[2];
f[0] = new Flower();
f[0].setColour("Red");
f[0].setHeight(4.0);
f[1].setColour("Blue");
f[1].setHeight(3.5);
```

(*) Use the **`Flower`** class defined in slides 28+29 of the "Object Basics: how OO works" topic in **Teaching Week 1**.

Question 4

- Assume we have a **`Flower`** class (*), and two of its methods are **`setColour(String colour)`** and **`setHeight(double height)`**. What is wrong with the following code? Explain your answer.

```
Flower[] f = new Flower[2];
f[0] = new Flower();
f[0].setColour("Red");
f[0].setHeight(4.0);
f[1] = new Flower();
f[1].setColour("Blue");
f[1].setHeight(3.5);
f[2] = new Flower();
f[2].setColour("Pink");
f[2].setHeight(2.5);
```

(*) Use the **`Flower`** class defined in slides 28+29 of the "Object Basics: how OO works" topic in **Teaching Week 1**.

Question 5

- What is the output of this program? Explain your answer.

```
public class Test_Q5 {
  public static void main(String[] args) {
    int i;
    i = i + 5;
    System.out.println("i = " + i);
  }
}
```

Question 6

- A class **`Square`** is defined as:

```
public class Square {
  public int square(int i) { return i*i; }
  public double square(double i) { return i*i; }
}
```

What is the output of the program below? Explain your answer.

```
public class SquareTest {
  public static void main (String[] args) {
    int i= 6;
    Square s = new Square();
    System.out.println(s.square(i));
    double x = i;
    System.out.println(s.square(x));
  }
}
```

Question 7

- Is this valid code? If it is invalid, explain what is the problem.

```
public class Square_Q7a {
  public int square(int x) { return x*x; }
  public double square(int y) { return y*y; }
}
```

```
public class Square_Q7b {
  public double square(int x) { return x*x; }
  public double square(double y) { return y*y; }
}
```

```
public class Square_Q7c {
  public double square(int x) { return x*x; }
  public int square(double y) { return y*y; }
}
```

Question 8

- Given the classes **`Car`** and **`Truck`** defined below, what is the output of the code fragment shown in the box?

```
Truck mycar = new Truck();
System.out.println(mycar);
mycar.m1();
mycar.m2();
```

```
public class Car {
   public void m1() { System.out.println("car 1"); }
   public void m2() { System.out.println("car 2"); }
   public String toString() { return "vroom"; }
}

    public class Truck extends Car {
       public void m1() { System.out.println("truck 1"); }
    }
```

Question 9

- Given the classes **`Car`** and **`Truck`** defined below, what is the output of the code fragment shown?

```
Truck mycar = new Truck();
System.out.println(mycar);
mycar.m1();
mycar.m2();
```

```
public class Car {
   public void m1(){ System.out.println("car 1"); }
   public void m2(){ System.out.println("car 2"); }
  public String toString() { return "vroom"; }
}

 public class Truck extends Car {
    public void m1() { System.out.println("truck 1"); }
   public void m2() { super.m1(); }
   public String toString() { return super.toString()+ "T"; }
 }
```

Question 10

- What is the output of this program? Explain your answer.

```
public class Test_Q10 {
  public static void main(String[] args) {
    String s = "6";
    int n = 3;
    double d = 4.5;
    System.out.println(s + n + d);
  }
}
```

Questions 11+12

- Identify which statements are TRUE and which are FALSE. Justify your answers.
  - ❑ A subclass has direct access to its superclass' private data and methods.
  - ❑ A class can extend more than one superclass.
  - ❑ An abstract class must contain at least one abstract method.
  - ❑ An abstract class must not contain any instance variables.

- A class **`Animal`** has a subclass **`Dog`**. Which of the following is TRUE?
  a) **`Dog`** cannot have subclasses.
  b) **`Dog`** has no other parent than **`Animal`**.
  c) **`Animal`** can have only one subclass.
  d) **`Dog`** cannot have siblings.

Question 13

- Determine the output of these programs.

```
public class Test_Q13a {
  public static void main( String[] args) {
    String s1 = new String("aaa");
    String s2 = new String("aaa");
    System.out.println(s1==s2);
  }
}
```

```
public class Test_Q13b {
  public static void main( String[] args) {
    String s1 = new String("aaa");
    String s2 = new String("aaa");
    System.out.println(s1.equals(s2));
  }
}
```

```
public class Test_Q10c {
  public static void main( String[] args) {
    String s1 = "aaa";
    String s2 = "aaa";
    System.out.println(s1==s2);
  }
}
```

Question 14

- Using the **`BankAccount`** class (*) as the superclass, write a class called **`CurrentAccount`**.
  - A **`CurrentAccount`** object, in addition to the attributes of a **`BankAccount`** object, should have an *overdraft limit* variable.
  - Override methods of the **`BankAccount`** class if necessary.
  - Now create a **`Bank`** class, an object of which contains an **`ArrayList`** of **`BankAccount`** objects; accounts in the list could be instances of the **`BankAccount`** class, or instances of the **`CurrentAccount`** class.
  - The **`Bank`** class requires methods for *opening* and *closing* accounts.
  - Write an **`update()`** method in the **`Bank`** class; it iterates through each account, and **`CurrentAccount`** objects get a letter sent if they are in overdraft.

(*) Use the **`BankAccount`** class defined in <u>slide 12</u> of the slide set "Object-Oriented Programming (Extra Example)" in **Teaching Week 1**.

Exercise: Fill in the Gaps

- Consider the incomplete Java program **Group**. It contains a method that adds up all values in an array.

```
public abstract class Group {
  public void printGroup(int[] array) {
    // code missing
  }
}
```

- Use the collection of statements on the right, together with some extra right brackets (**}**), to fill out the abstract **Group** class.

```
for (int i=0; i<array.length; i++)
System.out.println(sum);
int sum = 0;
sum += array[i];
```

- Then write a new concrete class **MyGroup** that extends **Group**, <u>and</u> also write a test class that invokes the **printGroup()** method with the values **{12,34,43,21}** and displays their sum.

```
public class TestingGroup {
  // code missing
}
```

Exercise: Predict the Question

- Determine the set of questions that should result in the following answers:

  What is the Question for this Answer?

  "It is a class that must be extended in order to be instantiated. "

  What is the Question for this Answer?

  "This is the method that is called when the object must be represented as a text value, or when an object is referred to, in a manner in which a string is expected."

  What is the Question for this Answer?

  "This allows you to treat a class like any class within its object hierarchy. For example, a `Student` could be treated like a `Person` object".

You should only attempt this exercise after watching the "Inheritance and Abstract classes" lecture recordings, scheduled for days 4+5 of Teaching Week 2.