Tutorial: Teaching Week 2

- Data types
- Arrays
- Random numbers
- Compiler & Runtime errors
- Inheritance

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?

```
  public class Test{
    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 **`setColour(String colour)`** and **`setHeight(double height)`** are two of its methods. What is wrong with the following code?

```
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 23+24 of the "Object Basics: how OO works" topic in **Teaching Week 1**.

Question 4

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

```
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 23+24 of the “Object Basics: how OO works” topic in **Teaching Week 1**.

Question 5

- What Java statements to generate the following:

1. A random integer between **5** and **25** (inclusive).

2. A random integer between **`n`** and **`m`** (inclusive), where **`n < m`**.

Question 6

- What is the output of this program?

```
public class Test {
  public static void main(String[] args) {
    int i;

    i = i + 5;
    System.out.println("i = " + i);
  }
}
```

Question 7

- 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 the output of the program below?

```
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 8

- Is this valid code?

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

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

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

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"); }
    }
```

Question 10

- 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 11

- What is the output of this program?

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

Question 12

- Identify which statements are TRUE and which are FALSE:
  - ❑ 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 {
  public static void main( String[] args) {
    String s1 = new String("aaa");
    String s2 = new String("aaa");
    System.out.println(s1==s2);
  }
}
```

```
public class Test2 {
  public static void main( String[] args) {
    String s1 = new String("aaa");
    String s2 = new String("aaa");
    System.out.println(s1.equals(s2));
  }
}
```

```
public class Test3 {
  public static void main( String[] args) {
    String s1 = "aaa";
    String s2 = "aaa";
    System.out.println(s1==s2);
  }
}
```

Queen Mary
University of London

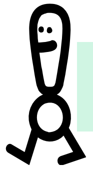
Question 14

- 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.

Question 15 – Extra

```
public abstract class HotelCost {
  protected int days;         // number of days booked
  protected double price;     // price per day
  public int getDays() { return days; }
  public double getPrice() { return price; }
  public void setDays(int n) { days = n; }
  public void setPrice(double p) { price = p; }
  public abstract double calCost(); // Calculate and return cost.
}
```

- Using the abstract class **`HotelCost`**, write a concrete sub-class called **`SimpleHotelCost`**. Add a new attribute **`tax`** and accessor (getter) / mutator (setter) for the **`tax`** variable. The **`calCost()`** method in **`SimpleHotelCost`** class simply calculates the cost using the formula **`days*price*(1+tax)`**.
- Write a test program for class **`SimpleHotelCost`**, that calls the **`calCost()`** method and displays the cost for variable values **`days=3`**, **`price=79.5`** and **`tax=0.2`**.



This is a past exam question.