Team Application Exercises (tAPP-7)

`Instructions:` Work on `problem 1` on your own for 5 minutes. Then discuss your code with your team members and present one solution as a team (5 minutes). Follow the same process for all the tasks. Swap your solutions with another team for peer evaluation (15 minutes).

Problem 1: This question involves information about phone calls made from a particular mobile phone over one calendar month. Details of calls are stored in a list, in which each element contains the number called and the duration of the call, represented as an instance of the **Call** class below:

```
public class Call {
    public String number;
    public int duration;
}
```

An example of a call list would be as follows:

[[“01234567890”, 5], [“09876543201”, 17], [“01234567890”, 14], . . .]

The following function is supposed to calculate the phone bill for the month, assuming that each call costs 0.5p per minute. The parameter **calls** is a list of **Call** objects in the format described above.

The code below contains **two errors**. Identify both errors, including how the error would affect the execution of the function, and say how to correct it.

```
public double calculateBill(List<Call> calls, String favourite) {
      Map<String, Integer> totals;
      for (Call call : calls) {
            totals = new HashMap<>();
            if (totals.containsKey(call.number)) {
                  totals.put(call.number,
                              totals.get(call.number) + call.duration);
            } else {
                  totals.put(call.number, call.duration);
            }
      }
      double totalCharge = 0;
      for (String number : totals.keySet()) {
            totalCharge += totals.get(number) / 2;
      }
      return totalCharge;
}
```

Error 1: totals is re-intialised inside the first **for** loop, so none of the values added to it will stay around. The initialisation should be moved outside the loop.

Error 2: the division inside the second **for** loop uses integer division, so will not compute the correct total cost. It should be converted to use floating-point division, for example by dividing by 2.0 instead of by 2.

Problem 2: Define an enumerated type **Operation** with values for the basic arithmetic operations (PLUS, MINUS, TIMES, DIVIDE). Your **Operation** type should also include a method **calculate(double x, double y)** that executes the relevant operation on its arguments – for example, **Operation.PLUS.calculate(2.0, 4.0)** should return 6.0.

```
public enum Operation {
	PLUS, MINUS, TIMES, DIVIDE;

	public double compute (double x, double y) {
		switch (this) {
		case PLUS:
			return x + y;

		case MINUS:
			return x - y;

		case TIMES:
			return x * y;

		case DIVIDE:
			return x / y;
		}
		// Need to have this so Java is sure that we return a value
		return 0;
	}
}
```

Problem 3: Define an interface **Figure** which should represent geometric shapes and should include two methods, **calculatePerimeter()** and **calculateArea()**, both of which should return a **double**. Then define three classes implementing **Figure** – **Circle**, **Triangle**, and **Rectangle** – each of which has an appropriate constructor that allows the two methods to work correctly (you can assume that the triangle is equilateral, with all sides the same length).

- Hint 1: for a circle of radius **r**, the perimeter is computed as 2 * π * r (use **Math.PI** to get the value of π), and the area is π * r * r
- Hint 2: for an equilateral triangle with side length **s**, the perimeter is 3 * s, and the area is (**Math.sqrt(3) / 4) * s * s**
- Hint 3: for a rectangle with width **w** and height **h**, the perimeter is 2 * (w + h) and the area is (w * h)

```
public interface Figure {
	double calculatePerimeter();
	double calculateArea();
}

public class Circle implements Figure {
	private int radius;

	public Circle (int radius) {
		this.radius = radius;
	}

	@Override
	public double calculatePerimeter() {
		return 2 * Math.PI * radius;
	}

	@Override
	public double calculateArea() {
		return Math.PI * radius * radius;
	}
}
```

```java
public class Triangle implements Figure {
	private double side;

	public Triangle(double side) {
		this.side = side;
	}

	@Override
	public double calculatePerimeter() {
		return 3 * side;
	}

	@Override
	public double calculateArea() {
		return (Math.sqrt(3) / 4) * side * side;
	}
}


public class Rectangle implements Figure {

	private double width;
	private double height;

	public Rectangle(double width, double height) {
		this.width = width;
		this.height = height;
	}

	@Override
	public double calculatePerimeter() {
		return 2 * (width + height);
	}

	@Override
	public double calculateArea() {
		return width * height;
	}

}
```