Programming 1 – live exercises

Exceptions

We have the following `CoffeeCup` class:

```java
public class CoffeeCup {
	/**
	 *  45 degrees Celsius: the best temperature for coffee
	 */
	public static final int OPTIMUM_TEMPERATURE = 45;
	private int temperature;

	public CoffeeCup(){
		this.temperature = OPTIMUM_TEMPERATURE;
	}

	public CoffeeCup(int temperature){
		this.temperature = temperature;
	}

	public int getTemperature(){
		return temperature;
	}
}
```

Task 1: Trigger exceptions

Declare a `Person` class that has two final class attributes:

i. Integer value indicating when a cup of coffee is considered too cold; initialise it with 30 degrees Celsius.
ii. Integer value indicating when a cup of coffee is considered too hot; initialise it with 60 degrees Celsius.

Also implement a `drinkCoffee` method that accepts as a parameter a `CoffeeCup` object and returns nothing. First, the method determines the temperature of the object passed and, depending on the given situation, the method triggers an exception, either with the information text "Coffee too cold!" or with the information text "Coffee too hot!".

Solution:

```java
public class Person {

    private static final int TOO_COLD = 30;
    private static final int TOO_HOT = 60;

    public void drinkCoffee(CoffeeCup cup) throws Exception {
        int temperature = cup.getTemperature();
        if (temperature <= TOO_COLD)
            throw new Exception("Coffee too cold!");
        if (temperature >= TOO_HOT)
            throw new Exception("Coffee too hot!");
        doSomethingElse();
        //...
            }

    private void doSomethingElse(){
        //TODO
    }
}
```

Task 2: Handling exceptions

a) Declare a `CoffeeApp1` application class which tests the two classes `Person` and `CoffeeCup`, regardless of the possibility of errors. In the test scenario, a person should drink a cup of coffee with a temperature of 25 degrees Celsius.

b) Now declare a `CoffeeApp2` application class that adequately handles the possible exception situations.

Solution a):

```java
public class CoffeeApp1 {

    public static void main(String[] args) throws Exception {
        Person p1 = new Person();
        p1.drinkCoffee(new CoffeeCup(25));

//      Person p2 = new Person();
//      p2.drinkCoffee(new CoffeeCup(80));

    }
}
```

Solution b):

```java
public class CoffeeApp2 {

    public static void main(String[] args) {
        Person p1 = new Person();
        try {
            p1.drinkCoffee(new CoffeeCup(25));
        } catch (Exception ex) {
            System.out.println("Drink coffee failed!");
        }
}
```

Task 3: User-defined exception classes

a) Declare two separate exception classes `TooColdException` and `TooHotException`, whereby each exception object can be given a suitable text.

Solution:

```
public class TooColdException extends Exception {

    public TooColdException() {
    }

    public TooColdException(String s) {
        super(s);
    }

}


public class TooHotException extends Exception {
    public TooHotException() {
    }

    public TooHotException(String message) {
        super(message);
    }

}
```

b) Then rewrite the `drinkCoffee` method of the `Person` class so that the customised exception class is used in the respective exception situation.

Solution:

```
public class Person {

      public static final int COLD_COFFEE = 30;
      public static final int HOT_COFFEE = 60;

      public void drinkCoffee(CoffeeCup cup) throws TooColdException,
TooHotException {

            int temp = cup.getTemperature();

            if(temp <= COLD_COFFEE){
                  throw new TooColdException("Coffee too cold!");
            }

            if(temp >= HOT_COFFEE){
                  throw new TooHotException("Coffee too hot!");
            }
            //drink

      }

}
```

c) Declare a `CoffeeApp3` application class which tests the two classes `Person` and `CoffeeCup`. In the test scenario, the first person should drink a cup of coffee with a temperature of 25 degrees Celsius and then a second person drinks a cup of coffee with a temperature of 80 degrees Celsius. In doing so, all possible exception situations should be handled appropriately.

Solution:

```
public class CoffeeApp3 {

      public static void main(String[] args) {
            try {

                  Person p1 = new Person();
                  CoffeeCup c = new CoffeeCup(25);
                  p1.drinkCoffee(c);
            } catch (TooColdException ex) {
                  System.out.println("Coffee is too cold");
                  ex.printStackTrace(System.out);
            } catch (TooHotException ex) {
                  System.out.println("Coffee is too hot");
                  ex.printStackTrace(System.out);
            }

            try {
                  Person p2 = new Person();
                  p2.drinkCoffee(new CoffeeCup(80));
            } catch (TooColdException ex) {
                  System.out.println("Coffee is too cold");
                  ex.printStackTrace(System.out);
            } catch (TooHotException ex) {
                  System.out.println("Coffee is too hot");
                  ex.printStackTrace(System.out);
            }
      }
}
```