BSCCS2005: Graded Assignment with Solutions
Week 7

1. Consider the code given below. [ARUP:MCQ:2 points]

```
import java.util.logging.*;

public class SomeClass {
    public void logIt(){
        Logger.getGlobal().info("First message");
    }
}

public class FClass {
    public static void main(String[] args){
        SomeClass obj = new SomeClass();
        obj.logIt();
        Logger.getGlobal().log(Level.FINE, "second message");
        Logger.getGlobal().setLevel(Level.OFF);
        try {
            throw new ArithmeticException();
        }
        catch(Exception e) {
            Logger.getGlobal().log(Level.SEVERE, "third message");
        }
    }
}
```

Identify the result when the code gets executed.

◯ It prints nothing

√
```
<date time> SomeClass logIt
INFO: First message
```

◯
```
<date time> SomeClass logIt
INFO: First message
<date time> FClass main
FINE: second message
```

◯
```
<date time> SomeClass logIt
INFO: First message
<date time> FClass main
SEVERE: third message
```

Solution: By default, the top three levels of the logging levels are logged. Thus, `second message` is not printed.
The statement `Logger.getGlobal().setLevel(Level.OFF);` suppress all the logging. Thus, the string `third message` is not printed.

2. Consider the Java classes (each written into a different file as denoted) given below.

[ARUP:MCQ:2 points]

```
//FILE-1
package in.ac.iitm;
import java.util.logging.*;
public class SomeClass{
    private final static Logger logbook = Logger.getLogger("in.ac.iitm");
    public void doIt(){
        logbook.warning("start of doIt() in in.ac.iitm");
        logbook.setLevel(Level.OFF);
        logbook.warning("end of doIt() in in.ac.iitm");
    }
}

//FILE-2
package in.ac.iitm.onlinedegree;
import java.util.logging.*;
public class SomeClass{
    private final static Logger logbook =
                            Logger.getLogger("in.ac.iitm.onlinedegree");
    public void doIt(){
        logbook.warning("start of doIt() in in.ac.iitm.onlinedegree");
        logbook.setLevel(Level.OFF);
        logbook.warning("end of doIt() in in.ac.iitm.onlinedegree");
    }
}

//FILE-3
public class FClass{
    public static void main(String[] args){
        in.ac.iitm.SomeClass obj1 = new in.ac.iitm.SomeClass();
        in.ac.iitm.onlinedegree.SomeClass obj2 =
                                    new in.ac.iitm.onlinedegree.SomeClass();

        obj1.doIt();
        obj2.doIt();
    }
}
```

Identify the result when the code gets executed.

√ `<date time> in.ac.iitm.SomeClass doIt`
`WARNING: start of doIt() in in.ac.iitm`

◯
```
<date time> in.ac.iitm.SomeClass doIt
WARNING: start of doIt() in in.ac.iitm
<date time> in.ac.iitm.onlinedegree.SomeClass doIt
WARNING: start of doIt() in in.ac.iitm.onlinedegree
```

◯
```
<date time> in.ac.iitm.SomeClass doIt
WARNING: start of doIt() in in.ac.iitm
<date time> in.ac.iitm.SomeClass doIt
WARNING: end of doIt() in in.ac.iitm
<date time> in.ac.iitm.onlinedegree.SomeClass doIt
WARNING: start of doIt() in in.ac.iitm.onlinedegree
```

◯
```
<date time> in.ac.iitm.SomeClass doIt
WARNING: start of doIt() in in.ac.iitm
<date time> in.ac.iitm.SomeClass doIt
WARNING: end of doIt() in in.ac.iitm
<date time> in.ac.iitm.onlinedegree.SomeClass doIt
WARNING: start of doIt() in in.ac.iitm.onlinedegree
<date time> in.ac.iitm.onlinedegree.SomeClass doIt
WARNING: end of doIt() in in.ac.iitm.onlinedegree
```

Solution: The program logger two different logger for two different `SomeClass` classes.
Since, logger names are hierarchical, if we set log level as `OFF` on the logger `in.ac.iitm`, then the child logger `in.ac.iitm.onlinedegree` inherits that level.

3. Consider the Java code given below. [ARUP:MCQ:2 points]

```
public class MainClass{
    public static double compute(int a, int b){
        int c = 0;
        assert a > 0: "a must be > 0";    //assert-1
        assert b > 0: b;                  //assert-2
        c =  a / b;
        assert c >= 0: c;                 //assert-3
        return Math.sqrt(c);
    }
    public static void main(String[] args){
        int a = 10;
        int b = -5;
        assert b != 0: "b == 0";          //assert-4
        compute(a, b);
    }
}
```

Identify the first `assert` statement that throws the `AssertionError` when the class is executed as:

```
java -ea MainClass
```

- ◯ assert-1
- √ assert-2
- ◯ assert-3
- ◯ assert-4

Solution: The condition given for the assert statement `assert-2` is false, so it throws the `AssertionError`.

4. Consider the Java code given below. [ARUP:MCQ:2 points]

```
public class DOBRegistration{
    private int day, month, year;
    public DOBRegistration(int day, int month, int year){
        assert 0 < day && day <= 31: "day :" + day;        //assert-1
        this.day = day;
        assert 0 < month && month <= 12: "day :" + day;    //assert-2
        this.month = month;
        this.year = year;
    }
}
public class JobApplication{
    private int age;
    public JobApplication(int age){
        assert age >= 18: "invalid age for job";            //assert-3
        this.age = age;
    }
}
public class TaxReturn {
    private double income;
    public TaxReturn(double income){
        assert income >= 100000.00: income;                 //assert-4
        this.income = income;
    }
}
public class FClass3{
    public static void main(String[] args){
        DOBRegistration dr = new DOBRegistration(2, 23, 1879);
        JobApplication ja = new JobApplication(20);
        TaxReturn tr = new TaxReturn(75000.00);
    }
}
```

Identify the `assert` statement that throws the `AssertionError` when the class is executed as:

```
java -ea:...  -da:DOBRegistration FClass
```

◯ assert-1

◯ assert-2

◯ assert-3

√ assert-4

Solution: Since assertions are enabled for all the classes exxcept `class DOBRegistration`. Thus, assert statement `assert-4` throws `AsserionError`.

5. Consider the following Java code and choose the correct option.

[Anand : MCQ : 2 points]

```
public class Example {
    public static void main(String[] args) {
        int a=10,b=0;
        try{
            int c=a/b;
            System.out.println("Quotient is "+c);
        }
        catch (Exception ae){
            System.out.println("Exception handled");
        }
        catch (ArithmeticException ae){
            System.out.println("ArithmeticException handled");
        }
    }
}
```

◯ This code generates the output:
`Exception handled`

◯ This code generates the output:
`ArithmeticException handled`

◯ This code generates the output:
`Exception handled`
`ArithmeticException handled`

√ Compilation error

Solution: catch blocks must be ordered from most specific exceptions to most general exception; otherwise the specific exception block after general exception block becomes unreachable code.

6. Consider the following Java code and choose the correct option. [Anand : MCQ : 2 points]

```
public class Example{
    public static void main(String[] args) {
        try{
            int a=10/0;
        }
        finally{
            System.out.println("In finally block");
        }
        System.out.println("Program execution finished");
    }
}
```

- ◯ Compilation error
- √ This program terminates abnormally after printing the message:
  `In finally block`
- ◯ The program terminates successfully after printing the message:
  `In finally block`
- ◯ The program terminates successfully after printing the message:
  `In finally block`
  `Program execution finished`

Solution: In the above program, there is no corresponding catch block for handling `ArithmeticException`, hence the program terminates abnormally.

7. Consider the following Java code and choose the correct option for `Line 1` such that the code prints: `String index out of its range`. [Anand : MCQ : 2 points]

```
public class Example {
    public static void main(String[] args) {
        String name = "IIT Madras";
        try{
            System.out.println(name.charAt(10));
        }
        //Line 1
    }
}
```

◯
```
catch (StringIndexOutOfBoundsException e){
  System.out.println("String index out of its range");
}
```

◯
```
catch (Exception e){
  System.out.println("String index out of its range");
}
```

◯
```
catch (Throwable t){
  System.out.println("String index out of its range");
}
```

√ All of the above

Solution: In the above program, the statement `name.charAt(10)` throws `StringIndexOutOfBoundsException` which can be caught by using StringIndexOutOfBoundsException/Exception/Thowable catch blocks.

8. Consider the following Java code and choose the correct option.

[Anand : MCQ : 2 points]

```
public class Example{
    public void show(){
        NullPointerException e = new NullPointerException();
        e.initCause(new ArithmeticException());
        throw e;
    }
    public static void main(String[] args) {
        Example object = new Example();
        try{
            object.show();
        }
        catch (Exception e){
            System.out.println(e);
            System.out.println(e.getCause());
        }
    }
}
```

- √ This program generates the output.
  `java.lang.NullPointerException`
  `java.lang.ArithmeticException`
- ◯ This program generates the output.
  `java.lang.NullPointerException`
- ◯ This program generates the output.
  `java.lang.ArithmeticException`
- ◯ Compilation error

Solution: In above program used exception rethrow concept.
Here java.lang.NullPointerException chained with the java.lang.ArithmeticException

9. Consider the following Java code and choose the correct option. [Anand : MCQ : 2 points]

```
public class Example{
    public static void main(String[] args) {
        int a=10,b=0;
        try{
            int c=a/b;
            System.out.println("Quotient is "+c);
        }
        catch (ArithmeticException e){
            System.out.println(10/0);
            System.out.println("b value should not be zero");
        }
        catch (Exception e){
            System.out.println("Exception handled");
        }
    }
}
```

- ◯ Compilation error
- ◯ The program terminates normally after printing the message: `b value should not be zero`
- ◯ The program terminates normally after printing the message: `Exception handled`
- √ The program terminates abnormally due to unhandled exception(s).

Solution: The statement `int c = a/b;` causes an `ArithmeticException`, which results in the execution of the corresponding catch block. However, the statement `System.out.println(10/0);` inside the catch block throws another `ArithmeticException`, which remains unhandled. As a result, the program gets terminated abruptly.

10. Consider the following Java code and choose the correct option(s). [Anand : MSQ : 2 points]

```
//Data.java
package util.iitm.java;
public class  Data{
    void show(){
        System.out.println("This is show");
    }
}

//UseData.java
package iitm.java.program;
public class UseData {
    public static void main(String[] args) {
        new util.iitm.java.Data().show();
    }
}
```

- ◯ Compilation error in `Data.java`
- √ `Data.java` gets compiled.
- √ Compilation error in `UseData.java`
- ◯ `UseData.java` gets compiled.

Solution: `Data.java` gets compiled without any errors and `Data.class` is created under the `util.iitm.java` package.
Compilation error in `UseData.java` because `show()` is not declared as `public`, and hence you cannot access it outside the package.

11. Consider the following two source code files located in two different packages as shown.

[MSQ : 2 points]

//Adder.java

```
package iitm;
public class Adder {
    int add(int n1, int n2, int n3) {
        return n1+n2+n3;
    }
    protected int add(int n1,int n2) {
        return n1+n2;
    }
}
```

//Test1.java

```
package test;
import iitm.*;

class Calculator extends Adder{
    public void calculate() {
        System.out.println(this.add(7,8,9)); // LINE 1
        System.out.println(this.add(9,10));  // LINE 2
    }
}
public class Test1{
    public static void main(String args[]) {
        Adder a1 = new Adder();

        System.out.println(a1.add(4,5));   // LINE 3
        System.out.println(a1.add(1,2,3)); // LINE 4

        new Calculator().calculate();
    }
}
```

Choose the correct option regarding these two .java files.

- ◯ LINE 1 will not lead to compilation error.
- √ LINE 2 will not lead to compilation error.
- ◯ LINE 3 will not lead to compilation error.
- ◯ LINE 4 will not lead to compilation error.

Solution: `protected` members of a class can only be accessed in subclasses within the package as well as outside the package.
Members of a class where access specifier is not mentioned explictly are treated as ***package-private*** types and can only be accessed within that specific package. `public` members are accessible through out all packages and all classes
Therefore the `Adder` object `a1` inside the `test1` class's `main` method can only access public members of `Adder` class.
The `Calculator` class can access the `protected` members of the `Adder` class along with `public` members (if any) because `Calculator` class is a subclass of `Adder`.