BSCCS2005: Graded Questions with Solutions
Week 9

1. Consider the following code and choose the correct option.

```
import java.util.*;
public class User{
   public String name;
   public String pass;
   User(String name,String pass){
      this.name=name;
      this.pass=pass;
   }
   String checkUser(){
      if(pass.equals("1234")){
         return name;
      }
      else{
         return pass;
      }
   }
}
public class Test{
   public static void main(String[] args){
      Optional<String> op1=Optional.ofNullable(new User(null,"1234").checkUser());
      Optional<String> op2=Optional.ofNullable(new User("Moon","4321").checkUser());
      System.out.println(op1.orElse("Sun")+"\n"+op2.orElse("Sun"));
   }
}
```

- ◯ This program generates output:
  Sun
  Sun
- ◯ This program generates output:
  1234
  Moon
- √ This program generates output:
  Sun
  4321
- ◯ This program generates output:
  Sun
  Moon

Solution: `ofNullable(T value)` method returns an `Optional` describing the specified value, if non-null, otherwise returns an empty `Optional`.
`orElse(T other)` method returns the value if present, otherwise returns `other`.

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

```
import java.util.*;
public class Example{
   Optional<Integer> count;
   Example(Optional<Integer> count){
      this.count=count;
   }
   void show(){
      Optional<Integer> num=count.filter(n->n>18);
      if(num.isPresent()){
         System.out.println(num.get());
      }
      else{
         System.out.println(count.map(n->18-n).get());
      }
   }
}
public class Test{
   public static void main(String[] args){
      Optional<Integer> op1=Optional.of(20);
      Optional<Integer> op2=Optional.of(10);
      new Example(op1).show();
      new Example(op2).show();
   }
}
```

◯ This program generates NullPointerException.

√ This program generates output:
20
8

◯ This program generates output:
20
10

◯ This program generates output:
18
8

Solution: `filter(Predicate<? super T> predicate)` method returns an `Optional` describing the value if a value is present and the value matches the given predicate, otherwise, returns an empty `Optional`.
`map(Function<? super T,? extends U> mapper)` method takes a mapping function as an input, which accept reference of type T and return U. If the value of T is null, `map()` returns empty `Optional`. If the mapping function is null or returns a null result, `map()` throws `NullPointerException`.

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

```
import java.util.*;
public class UserException extends RuntimeException {
   public UserException(String str){
      super(str);
   }
}
public class Example{
   Optional<Integer> op;
   Example(Optional<Integer> op){
      this.op=op;
   }
   void show(){
      System.out.println(op.orElseThrow(()->
                new UserException("No number detected")));
   }
}
public class Test{
   public static void main(String[] args){
      Optional<Integer> op=Optional.of(100);
      Optional<Integer> op2=Optional.empty();
      new Example(op).show();
      new Example(op2).show();
   }
}
```

- √ This program prints 100 followed by UserException with message "No number detected".
- ◯ This program generates NullPointerException .
- ◯ This program generates output:
  100
  null
- ◯ This program generates output:
  100
  Optional.empty

Solution: `orElseThrow(Supplier<? extends X> exceptionSupplier)` returns the contained value, if present, otherwise throw an exception created by the provided `supplier`.

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

```
import java.util.*;
import java.util.stream.*;

public class Test{
    public static void main(String[] args){
        var list=new ArrayList<Integer>();
        for(int i=10;i>0;i--){
            list.add(i);
        }
        IntSummaryStatistics stat = list.stream().
                collect(Collectors.summarizingInt(x->x));
        System.out.println(stat.getAverage()+"\n"+
                stat.getSum()+"\n"+stat.getMax()+"\n"+stat.getMin());
    }
}
```

√ This program generates output:
5.5
55
10
1

◯ This program generates output:
5
55
10
1

◯ This program gives compile time error because list cannot be converted to stream.

◯ This program generates compile time error with message "incompatible types: possible lossy conversion from double to int".

5. Consider the code given below.

```
import java.util.*;
import java.util.stream.*;
public class StreamExample2 {
    public static void main(String[] args) {
        var fruits = new ArrayList<String>();
        fruits.add("Bananas");
        fruits.add("Apples");
        fruits.add("Kiwis");
        fruits.add("Grapes");
        fruits.add("pomegranates");
        fruits.add("watermelons");
        Stream<String> fruitstream=fruits.stream();
        IntSummaryStatistics summary =
                fruitstream.collect(Collectors.summarizingInt(String::length));
        double avgLengthFruit = summary.getAverage();
        int maxLengthFruit = summary.getMax();
        System.out.println(avgLengthFruit);
        System.out.println(maxLengthFruit);
    }
}
```

Choose the correct option regarding the code.

√ This program generates output:
```
7.833333333333333
12
```

◯ This program generates output:
```
0
12
```

◯ This program generates a compilation error.

◯ This program terminates abnormally due to unhandled exception(s).

Solution: getAverage() this method will return the average string length of the all the strings in `fruits` object.
getMax() this method will return the length of the maximum length string among all the strings in `fruits` object.

6. Consider the code given below.
Assume that the file "E:\\Files\\java.txt" contains the following text.
Java is a programming language.

```
import java.io.*;
public class Example {
    public static void main(String[] args) {
        try {
            var in=new FileInputStream("E:\\Files\\java.txt");
            var din=new DataInputStream(in);
            System.out.println(din.read());
            System.out.println(din.readLine());
        }
        catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Choose the correct option regarding the code.

- ◯ This program generates output:
  J
  Java is a programming language.
- ◯ This program generates output:
  74
  Java is a programming language.
- √ This program generates output:
  74
  ava is a programming language.
- ◯ This program generates a compilation error.

Solution: read() methods read only one character and will return ASCII value of that character.
readLine() method reads an entire line from the file.

7. Consider the code given below.
Assume that there is no file named "E:\\Files\\tree.txt".

```
import java.io.*;
public class Example {
    public static void main(String[] args) {
        try {
            var out=new FileOutputStream("E:\\Files\\tree.txt");
            var dout=new DataOutputStream(out);
            String data="Trees give us fresh air.";
            dout.writeBytes(data);
            System.out.println("Data written to file successfully");
        }
        catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        catch (IOException e) {
            e.printStackTrace();
        }
        finally {
            System.out.println("Program execution finished.");
        }
    }
}
```

Choose the correct option regarding the code.

- ◯ This program generates compilation error.
- ◯ This program terminates abnormally due to unhandled exception(s).
- √ This program generates output:
  `Data written to file successfully`
  `Program execution finished.`
- ◯ This program terminates abnormally after printing the message:
  `Program execution finished.`

Solution: While writing data into the file, if such a file does not exist, then it will be created and data will be written into it.

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

```
import java.io.*;
public class Example{
    String str;
    int x;
    Example(String str, int x){
        this.str = str;
        this.x = x;
    }
    public String getStr(){
        return str;
    }
    public int getX(){
        return x;
    }
}
public class Test{
    public static void main(String[] args) throws Exception{
        ObjectOutputStream os =
                new ObjectOutputStream(new FileOutputStream("File.ser"));
        os.writeObject(new Example("Moon",1));
        FileInputStream fis=new FileInputStream("File.ser");
        ObjectInputStream ois=new ObjectInputStream(fis);
        Example e=(Example)ois.readObject();
        System.out.print(e.getStr()+"\n"+e.getX());
    }
}
```

◯ This program generates compile time error because class `Example` does not implement `Serializable`.

√ This program compiles successfully but throws `NotSerializableException` at runtime.

◯ This program generates output:
Moon
1

◯ This program generates output:
1
Moon

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

```
import java.io.*;
public class Example implements Serializable{
    transient String str="Moon";
    transient final int x=1;
}
public class Test{
    public static void main(String[] args) throws Exception{
        ObjectOutputStream os = new ObjectOutputStream(new FileOutputStream
                                ("File.ser"));
        os.writeObject(new Example());
        FileInputStream fis=new FileInputStream("File.ser");
        ObjectInputStream ois=new ObjectInputStream(fis);
        Example e=(Example)ois.readObject();
        System.out.print(e.str+"\n"+e.x);
    }
}
```

◯ This program generates output:
null
0

√ This program generates output:
null
1

◯ This program generates output:
0
1

◯ This program compiles successfully but throws `NotSerializableException` at runtime.

Solution: Once a variable is declared `final`, at runtime it is no longer in variable form but in value form. As `final` variable participates in serialization in value form and not in variable form therefore JVM can not check for transient. Hence there is no impact of transient on `final` variable.

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

```
import java.io.*;
import java.util.ArrayList;
public class Employee implements Serializable{
    transient String empName;
    static String company="ABC limited";
    double salary;
    public Employee(String empName, double salary) {
        this.empName = empName;
        this.salary = salary;
    }
}
public class SerialTest1{
    public static void main(String[] args) throws Exception{
        var fos=new FileOutputStream("Employee.txt");
        var os=new ObjectOutputStream(fos);
        ArrayList<Employee> list=new ArrayList<Employee>();
        list.add(new Employee("John",10000.00));
        list.add(new Employee("Jock",20000.00));
        os.writeObject(list);
        var fis=new FileInputStream("Employee.txt");
        var ois=new ObjectInputStream(fis);
        ArrayList<Employee> empList=(ArrayList<Employee>)ois.readObject();
        for(Employee emp:empList){
            System.out.print(emp.empName+" "+emp.salary+" "+emp.company);
        }
    }
}
```

◯ Compilation error.

◯ The program terminates abnormally during the deserialization process.

◯ The program terminates abnormally during the serialization process.

√ This program generates output:

```
null 10000.0 ABC limited
null 20000.0 ABC limited
```

Solution: