RAJARATA UNIVERSITY OF SRI LANKA

FACULTY OF APPLIED SCIENCES

B.Sc. (General) Degree

Second Year – Semester I Examination – September/October 2013

COM 2305 – JAVA PROGRAMMING LANGUAGE

Answer **All** questions. Time allowed: 3 hours

---

1 (a) Explain what you understand by the term "widget" and list down four (04) examples for "widgets". [1+2]

(b)
```
import javax.swing.*;
public class A
{
public static void main(String[] args)
{
JFrame frame = new JFrame("My First Frame");
}
}
```
Above is a small program to create a Frame (Window). If you run the above program it will not throw any exceptions. But you will not see a Frame (Window) either. Write the code segment required to solve the problem. [2]

(c) Give a brief description on "**swing.JfileChooser**" explaining its uses. [2]

(d) Write down the output of the following program.
```
class Test_One
{
    public static void main(String[] args)
    {
        int a = 10;
        Double b = 10.7324;
        Integer c = 010;
        int d = 0x10;
        int e = 0b10;
```

```
        System.out.println(a+b);
        System.out.println(a+c);
        System.out.println(a+d);
        System.out.println(a+e);
        System.out.println(b/a);
        System.out.println(b+e);
    }
}
```

(e) The character 'a' is represented by the integer value 97. Write the code segment to print the english alphabet (in lower case) programmatically using a "for" loop. Hint: use type casting. [3]

(f) Read the following. [3]

To determine whether a year is a leap year, follow these steps:
1. If the year is divisible by 4, go to step 2. Otherwise, go to step 5.
2. If the year is divisible by 100, go to step 3. Otherwise, go to step 4.
3. If the year is divisible by 400, go to step 4. Otherwise, go to step 5.
4. The year is a leap year (it has 366 days).
5. The year is not a leap year (it has 365 days).

```
boolean checkLeapYear(int year){
// code
}
```

checkLeapYear(int year) methods will get the "year" as an int and will return whether it (the "year") is a leap year or not (true for leap year and false otherwise). Complete the method (write the code inside the method).

(g) The above *checkLeapYear()* method takes a int as the input. Now consider code segment bellow (Assume *checkLeapYear()* is in *TestTwo* class). [5]

```
public static void main(String[] args)
{
Integer X = 1992;
TestTwo ae = new TestTwo();
System.out.println(ae.checkLeapYear(i));
}
```

Assume that this compiles and runs without an exception. You may notice that "i" (Integer type) is passed to the *checkLeapYear* method, even though *checkLeapYear*() has int as a parameter. Briefly explain the transformation which made this (successful compilation and running) possible.

2 (a) Briefly explain the difference between primitive types and reference types and write an [2]

example for each.

[2]

(b)
```
class Test_Two
{
public static void main(String[] args)
{
System.out.println(( 7 / 2.0 ) *2.0 );
System.out.println(( 7 / 2 ) * 2.0 );
System.out.println(( 7 / 2 ) * 2 );
System.out.println( 19.7 % 4 );
System.out.println(4 & 5);

System.out.println(!(!true &  (!false)));
System.out.println(!true | !(false & !false));

args = new String[3];
System.out.println(args[2]);

Integer a = 23;
int b = a.compareTo(32);

System.out.println(b);
System.out.println(!(5.2001 <= 5.2)?!false:!true);
}
}
```

List down the output of the above program.

[10]

(c) Name the parameter/s of the main method.

[1]

(d) Explain what operators are in java and give example each for urinary, binary and ternary operators.

[3]

(e) Write code segment to obtain a pseudo-random number (does not need to be an integer) between 0-1000.

[2]

(f) Compare and contrast local variables and parameters.

[1+1]

3 (a)
```
class Test_Three
{
int i = 5;
String s = "s";
```

```
    Test_Three()
    {
        System.out.println(this.s);
    }
    Test_Three(int i)
    {
        this(i,"a");
        System.out.println(i);
    }
    Test_Three(int i, String s)
    {
        this();
        System.out.println("i"+this.s);
    }
    Test_Three(String s, int i)
    {
        this(i);
        System.out.print(s+i);
    }
    public static void main(String[] args)
    {
        new Test_Three("A",2);
    }
}
```

List down the output of the above program. [4]

(b) Write down the output of the following program.

```
class TF
{
    static int i;
    String k;
}

class Test_Four
{
    int i = 5;
void say( )
{
    System.out.println(TF.i);
}

void say(int i)
{
    i = 2;
    System.out.println(i);
}
```

```
void say(TF tf)
{
        tf = new TF();
        System.out.println(tf.i);
        System.out.println(tf.k);
        tf.i = 10;
}

public static void main(String[] args)
        {
                int i = 1;
                int k = 2;
                Test_Four tf = new Test_Four();
                tf.say(i);
                System.out.println(i);

                TF.i = i;
                tf.say();

                TF t = new TF();
                t.k = "TF";
                tf.say(t);
                System.out.println(t.i);
                System.out.println(TF.i);
                System.out.println(t.k);

        }
}
```

[8]

(c) Above class have multiple methods with the same name. Write the name of this feature in Java.
Explain why having multiple methods with the same name didn't produce exception in the above class (**Test_Four**).

[3]

(d) If you need to declare method that would accept an arbitrary number of values (of a certain type) to a method, name and describe (with examples) the construct you are going to use.

[2]

(e) What will happen if a class does not have a constructor defined within it?

[2]

(f)
```
class Test_Five
{
  static int i = 5;
```

```
    static void foo()
    {
            System.out.println(this.i);
    }

    public static void main(String[] args)
    {
            Test_Five.foo();
    }
}
```

This would produce an exception. Briefly explain why?

[1]

4 (a)
```
class Test_Seven_Super {
void say() { System.out.println("super"); }
static void name(){ System.out.println("Test_Seven_Super"); }
}
class Test_Seven extends Test_Seven_Super {
void say(){ System.out.println("sub"); }
static void name(){ System.out.println("Test_Seven"); }
public static void main(String[] args){
Test_Seven_Super ts = new Test_Seven();
ts.say();
ts.name();
}}
```

List down the output of this program.

[2]

(b) Assume that **A**, **B**, **C** and **D** are classes in the exception hierarchy. Their locations in the hierarchy can be shown as A ← D ← C ←B.
(Parent class ← child class)

```
try{
File = getFile(new FileLocation(s));
}catch( (1)…… )}//code
}catch( (2)…… )}//code
}catch( (3)…… )}//code
}catch( (4)…… )}//code
}
```

Above getFile() throws all four (4) that **A**, **B**, **C** and **D** exceptions. Write the order in which exceptions should be handled.

[2]

(c)
```
class Test_Eight
{
```

```
void add(int size)
{
if (size == 0)
{
throw new ArrayIndexOutOfBoundsException();
}
}
public static void main(String[] args)
{
new Test_Eight().add(0);
}
}
```

Will this class compile without an error?

[1]

(d) Consider the following java classes

```
class Thread_One extends Thread {
    public void run() {
      for(int i=0;i<10;i++)System.out.print("*");
    }
}

class Thread_Two implements Runnable {
   public void run(){
     for(int i=0;i<10;i++)System.out.print("+");
   }
}

class Test_Thread {
   public static void main(String[] args) {
   }}
```

Write the code segment to create (and start) **Thread_One** and **Thread_Two** threads inside the **main** method of **Test_Thread**.

[3]

(e) You need to sleep each thread for 1000 ms after each iteration (of the **for** loop). Make the necessary changes and re-write the above three classes.

[4]

(f) Write the code segment to check the priority (and print it) of the **Thread_One** and **Thread_Two** threads discussed in Q4 (a), (b), (c) and (d).

[2+1]

(g) Write the code segment to set the priority of the **Thread_One** and **Thread_Two** threads discussed above. Set the maximum priority to **Thread_One** and minimum priority to **Thread_Two**.

[3]

(h) What will the output of the above (f) program will look like (after the priorities and sleep time have been set)?

[2]

5 (a) Dr.Ian has developed new algorithm for enhancing images. He has patented it and is hoping to sell it as a package. But he is not hoping to develop a complete software tool using the algorithm. Rather he would sell only the algorithm as a class library (and API documentation) without any GUIs so he can further improve the algorithm. The buyers of the class library can use it in their image processing application and tools. The class library is developed using java. The content (codes) is hidden from the buyers and they are not allowed to change the functionality in the package. Discuss two methods that can be used to ensure that buyers can't access the classes in the package in unnecessary ways.

[4]

(b) Briefly discuss a situation where **finally** block (in exception handling) may not execute.

[1]

(c) Name two (2) situations which might cause an **Error** (Error which is discussed under exception handling) in a java program.

[1]

(d) Name the term used to describe the situation of a superclass class (static) method when it is overridden by a subclass class (static) method.

[1]

(e) What will happen if you override a superclass field with a subclass field?

[1]

(f) Why is it important to immediately close an I/O stream after using it?

[2]

(g) "Java Locks are Reentrant". Explain what you understand by this statement. State whether you agree or not with that statement.

[2]

(h) Can you define a constructor inside an abstract class? Can you create an object from an abstract class?

[2]

(i) Can a running Thread be garbage collected? Explain your answer.

[2]

(j) Write a small program to take an integer as a keyboard input decide whether that number is odd or even and print the result. Hint: Use Scanner class and next() method. Remember to convert the string to an integer.

[4]