King Fahd University of Petroleum & Minerals

**Information and Computer Science Department**

**ICS 201 – Introduction to Computing II**

First Semester 2016-2017 (161)

Midterm Time allowed: 120 minutes

Saturday, November 26th, 2016

|  |  |
| --- | --- |
| ***Name:*** |  |

|  |  |
| --- | --- |
| ***ID#:*** |  |

***Please circle your section number below***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Section | 01 |  |  | 02 |
| Instructor | Dr. Tarek |  |  | Dr. Sami |
| Day and Time | UTR08–08:50 |  |  | UTR09–09:50 |

**Notes:**

|  |  |  |
| --- | --- | --- |
| Question # | MaximumMark | ObtainedMark |
| 1 | 25 |  |
| 2 | 15 |  |
| 3 | 20 |  |
| 4 | 20 |  |
| 5 | 20 |  |
| Bonus | 5 |  |
| **Total** | **100** |  |

Good Luck

**Question 1 [25 Points (20 + 5)]**

**a) What is the right answer?**

* 1. What does a derived class automatically inherit from the base class?
		1. instance variables
		2. static variables
		3. public methods
		4. all of the above
	2. If the final modifier is added to the definition of a method, this means:
		1. The method may be redefined in the derived class.
		2. The method may be redefined in the sub class.
		3. The method may not be redefined in the derived class.
		4. None of the above.
	3. Java does not use late binding for methods marked as:
		1. final
		2. static
		3. private
		4. all of the above
	4. A class that implements an interface but only gives definitions for some of the method headings given in the interface is called a/an:
		1. concrete class
		2. abstract class
		3. discrete class
		4. friendly class
	5. How can you prevent a class from being extended?
1. Declare a class static.
2. Declare a class private.
3. Declare a class protected.
4. Declare a class final.
	1. Which of the following is false?
5. An abstract class can have instance variables and non-abstract methods
6. A subclass should implement all abstract methods or itself declared as abstract
7. References of an abstract class can be declared, but they should refer to an object of the non- abstract subclass
8. None of the above
	1. Which of the following is true?
9. A child class can extend a parent or implement an interface, but not do both.
10. A child class can extend just one parent and can implement just one interface
11. A child class can extend just one parent and can implement zero or more interfaces
12. A child class can extend zero or more parents, and can implement zero or more interfaces
	1. Which of the following is true?
13. An abstract class cannot have any final method
14. A final class cannot have any abstract method
15. An abstract method can be declared private
16. A public static method can be overridden
	1. Assume that class A extends class B, which extends class C. Also all the three classes implement the method test(). How can a method in a class A invoke the test() method defined in class C (without creating an object from class C).
17. test();
18. super.test();
19. super.super.test();
20. It is not possible to invoke test() method defined in C from a method in A.
	1. Can an abstract parent class have non-abstract children?
21. No--an abstract parent must have only abstract children.
22. No--an abstract parent must have no children at all.
23. Yes--all children of an abstract parent must be non-abstract.
24. Yes--an abstract parent can have both abstract and non-abstract children.

**b) What is the output of the following TestClass:**

public class Father

{

 int y = 10;

 public Father(int y)

 {

 this.y = y;

 System.out.println(y);

 System.out.println("Father");

 }

 public Father()

 {

 this(5);

 System.out.println(y);

 play();

 }

 public void play()

 {

 System.out.println("I'm grown up, i don't play!");

 System.out.println(y);

 }

}

Public class Child extends Father

{

 int x = 10;

 public Child(int x)

 {

 this.x = x;

 System.out.println("Child");

 System.out.println(x);

 play();

 }

 public void play()

 {

 System.out.println("Let's play!");

 System.out.println(x);

 }

}

Public class TestClass

{

 public static void main(String[]args)

 {

 Child c = new Child(7);

 }

}

**Answer**

**Question 2 [15 Points]**

1. If a method includes a try block with multiple catch blocks, is the order of placing the catch blocks important? If the answer is no, explain why. If the answer is yes give an example where the order makes a difference.
2. Can a throw instruction be placed outside a try block? If the answer is no, explain why. If the answer is yes, describe an example of such situation.
3. When a java source file (.java) is compiled, to which format java instructions are converted?
4. What is the part of the JVM responsible for storing runtime objects?
5. What is the part of the JVM where instructions are executed?

**Question 3 [20 points]**

Write a GUI program in java to convert between temperatures from Centigrade to Fahrenheit and vice-versa. The GUI for the program should look like the following window:



Initially “C to F” radio button (*JRadioButton*) will be selected but the user can select any one of the two. If the user selects “C to F” and presses convert button (*JButton*), the program will expect a temperature value from user in the left text field (*JTextField*) i.e. for C, and will output the equivalent temperature in F on the right text field. If the user selects the “F to C” radio button and presses convert button, the program will expect a temperature value from the user in the right text field and will output the equivalent temperature in C on the left text field. The program should handle common user exceptions during input like if he enters a text instead of a number for the temperature. It should report the problem in a message box. The equation for temperature conversion is:

**C/5 = (F- 32) / 9**

**Question 4 [20 Points]**

4.1 Is it possible to declare the following sentence in java? **[2 Points]**

class Test extends Foo implements Comparable, Serializable, Formattable {

....

}

 4.2: What is wrong with the following interface? **[2 Points]**

public interface SomethingIsWrong {

 void aMethod(int aValue) {

 System.out.println("Hi Mom");

 }

}

4.3: Is the following interface valid? **[2 Points]**

public interface Marker {

}

4.4: Is there anything wrong with this exception handler as written? Will this code compile? **[3 Points]**

try {

} catch (Exception e) {

 } catch (ArithmeticException a) {

 }

4.5: Is the following code legal? **[3 Points]**

try {

} finally {

}

4.6: Modify the following cat method so that it will compile: **[8 Points]**

public static void cat(File file) {

 RandomAccessFile input = null;

 String line = null;

 try {

 input = new RandomAccessFile(file, "r");

 while ((line = input.readLine()) != null) {

 System.out.println(line);

 }

 return;

 } finally {

 if (input != null) {

 input.close();

 }

 }

}

**Question 5 [20 Points]**

5.1: The following java program will output **[2 Points]**

class A{

 int A1 = 70;

 String A2 = "AA";

 public A (int A1) {

 this("AB");

 System.out.println("A1 = " + A1 + ", A2 = " + A2);

 }

 public A(String A2){

 this.A1 = 80;

 this.A2 = A2;

 }

}

class B extends A{

 public B(){

 super(90);

 }

}

public class C{

 public static void main(String[] args){

 A What = new B();

 }

}

* 1. A1 = 70, A2 = AA
	2. A1 = 80, A2 = AB
	3. A1 = 90, A2 = AA
	4. A1 = 90, A2 = AB

5.2: Given the definition of the following classes, print the output produced by the main method in the Test class. **[2 Points]**

class Base {

 public Base() {

 this("Base(String s)");

 System.out.println("Base()");

 }

 public Base(String s) {

 System.out.println(s);

}}

class Child extends Base {

 public Child() {

 this(4775);

 System.out.println("BChild");

 }

 public Child(int value) {

 super("Exam 1");

 System.out.println("BChild(int value)");

 System.out.println(value);

}}

class GrandCh extends Child {

 public GrandCh(String a) {

 System.out.println(a);

}}

class Test {

 public static void main(String args[]) {

 new GrandCh ("GrandCh Created");

 }}

5.3: What is the output of the following program? **[2 Points]**

class G{

 int gr = 60;

 public G () {}

 public G (int gr) { this.gr=gr; ShowMe(); }

 public void ShowMe() {System.out.println("In G: gr = "+gr);}

}

class F extends G{

 int fa = 30;

 public F () {

 ShowMe();

 }

 public F (int gr) {

 super(gr);

 ShowMe();

 }

 public void ShowMe() { System.out.println("In F: gr ="+gr+" fa = "+fa);}

}

public class Major1{

 public static void main(String[] args){

 G g1 = new G(50);

 G f1 = new F(40);

 }

}

5.4: In this problem, you are asked to propose a set of classes organized in a class hierarchy. You are free to choose the application (i.e. geometrical shapes hierarchy, Employees hierarchy, Cars hierarchy, etc.). You will show how you can implement these classes using OOP concepts (i.e. encapsulation, inheritance, polymorphism, and abstraction). The proposed classes should manipulate information regarding the items in the hierarchy. **[14 Points, distributed as following]**

5.4.1: Draw the class hierarchy that shows the parent-child relationship between the proposed classes in your selected application. You need to have at least 4 classes **[2 Points]**

5.4.2: Write a Java program using of the above class hierarchy in which you need to demonstrate how you implemented the following OOP concepts (***i.e. encapsulation, inheritance, polymorphism, and abstraction***). **[6 Points]**

5.4.3: You need to explain the main advantages of using each of these concepts in your coding and how your program looks like if you did not use each of them. **[3 Points]**

5.4.4: You need to explain also what does it mean? Moreover, you need to highlight the corresponding code of each concept in your coded program if possible. **[3 Points]**

* Overriding,
* Overloading,
* Hiding,
* Shadowing,
* Early/static/compile time binding,
* Late/dynamic/runtime binding?

**Bonus Question [5 points]**

Write the code for the following method TwoSum such as:

Given an array of integers a and a target value, return the indices of two numbers inside the array such that they add up to the target value. The two indices should be returned in an array of size 2.

You may assume that each input would have exactly one solution.

TwoSum({2, 7, 11, 15}, 9) --> {0,1} (because a[0] + a[1] = 9)

TwoSum({8, 10, 11, 1, 4}, 15) --> {2,4} (because a[2] + a[4] = 15)

TwoSum({2, 7, 11, 15}, 9) --> {0,1} (because a[0] + a[1] = 9)

Public static int [] TwoSum(int[] a, int)

{

}