



**King Fahd University of Petroleum & Minerals**  
**College of Computer Sciences and Engineering**  
**Information and Computer Science Department**  
**Second Semester 102 (2010/2011)**

**ICS 201 - Introduction to Computing II**

Major Exam 1  
Wed., 23<sup>rd</sup> March, 2011  
Time: 120 minutes

*Name:*

*ID#:*

*Please circle your section number below:*

<i>Section</i>	01	02	03	04
<i>Instructor</i>	Sami	Tarek	Irfan	Sukari
<i>Day and Time</i>	SMW 9-9:50	SMW 8-8:50	SMW 0-10:50	SMW 13:00-13:50

Question #	Maximum Mark	Obtained Mark
1	30	
2	30	
3	10	
4	30	
<b>Total</b>	<b>100</b>	

**Question 1.1 [10 Points, one point each]**

- 1) What does a derived class automatically inherit from the base class?
  - (a) instance variables
  - (b) static variables
  - (c) public methods
  - (d) all of the above
  
- 2) If the final modifier is added to the definition of a method, this means:
  - (a) The method may be redefined in the derived class.
  - (b) The method may be redefined in the sub class.
  - (c) The method may not be redefined in the derived class.
  - (d) None of the above.
  
- 3) A class that implements an interface but only gives definitions for some of the method headings given in the interface is called a/an:
  - (a) concrete class
  - (b) abstract class
  - (c) discrete class
  - (d) friendly class
  
- 4) An inner class created inside a method can access
  - (a) Any local variables of the method where the class is created.
  - (b) Only instance variables of the enclosing class.
  - (c) Final local variables and all the variables (instance and static) of the enclosing class.
  - (d) Final local variables and instance variables of the enclosing class only.
  
- 5) How can you prevent a class from being extended?
  - (a) Declare a class static.
  - (b) Declare a class private.
  - (c) Declare a class protected.
  - (d) Declare a class final.
  
- 6) Which of the following is false?
  - (a) An abstract class can have instance variables and non-abstract methods
  - (b) A subclass should implement all abstract methods or itself declared as abstract
  - (c) References of an abstract class can be declared, but they should refer to an object of the non- abstract subclass
  - (d) None of the above
  
- 7) Which of the following is true?
  - (a) A child class can extend a parent or implement an interface, but not do both.
  - (b) A child class can extend just one parent and can implement just one interface
  - (c) A child class can extend just one parent and can implement zero or more interfaces

- (d) A child class can extend zero or more parents, and can implement zero or more interfaces
- 8) Which of the following is true?
- (a) An abstract class cannot have any final method
  - (b) A final class cannot have any abstract method
  - (c) An abstract method can be declared private
  - (d) A public static method can be overridden
- 9) 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).
- (a) test();
  - (b) super.test();
  - (c) super.super.test();
  - (d) It is not possible to invoke test() method defined in C from a method in A.
- 10) Can an abstract parent class have non-abstract children?
- (a) No--an abstract parent must have only abstract children.
  - (b) No--an abstract parent must have no children at all.
  - (c) Yes--all children of an abstract parent must be non-abstract.
  - (d) Yes--an abstract parent can have both abstract and non-abstract children.

-----  
**Question 1.2:** Follow the following program structure, fill the definition for missing sections (**commented //ADD YOUR CODE**) then write out the expected output. [20 Points]

```
public abstract class Shape {
    private static int count;
    public final double PI = (double)22/7;
    public Shape() { //ADD YOUR CODE

}

    public String toString(){
        return("This is a shape\n"+ "It is a " + getName());
    }
    abstract String getName();
}
```

```
public abstract class PlaneShape extends Shape {
    private double area;
    private double perimeter;
    public PlaneShape() { //ADD YOUR CODE

}
}
```

```
public double getArea(){
    return area;
}
public void setArea(double a){
    area = a;
}
public double getPerimeter(){
    return perimeter;
}
public void setPerimeter(double p){
    perimeter = p;
}
public String toString(){ //ADD YOUR CODE

}
abstract void calculateArea();
abstract void calculatePerimeter();
abstract String getName();
}

public class Square extends PlaneShape {
    private double side;
    public Square() { //ADD YOUR CODE

}

public void setSide(double s){ //ADD YOUR CODE

}

void calculateArea(){
    setArea(side*side);
}
void calculatePerimeter(){
    setPerimeter(4*side);
}
String getName(){
    return("Square");
}
public String toString(){ //ADD YOUR CODE

}
}
```

```
public class Circle extends PlaneShape{
    private double radius;
    public Circle() //ADD YOUR CODE

}
    public void setRadius(double r){ //ADD YOUR CODE

}
    void calculateArea(){
        setArea(PI*radius*radius);
    }
    void calculatePerimeter(){
        setPerimeter(2*PI*radius);
    }
    String getName(){
        return("Circle");
    }
    public String toString(){ //ADD YOUR CODE

}
}
```

```
public class AbstractClassQuestion {
    public static void main(String[] args) {
        Square s1 = new Square();
        s1.setSide(4);
        Circle c1 = new Circle();
        c1.setRadius(7);

        System.out.println(s1);
        System.out.println(c1);
    }
}
```

Program Output



**Question 3.1: [20 Points]** consider the following two classes “Person and Employee”:

```
public class Person {
    String firstName;
    String lastName;

    public Person(String f, String l)
    {   firstName = f;
        lastName = l; }

    public String toString()
    {   return "Person: " + firstName + " " + lastName; }

    public static void printDetails()
    {   System.out.println("This is Person " + lastName); }

    public Person generateSon()
    { return new Person(firstName+" Jr",lastName); }
}

public class Employee extends Person{
    double salary;

    public double getSalary()
    { return salary; }

    public String toString()
    {return "Employee: " + firstName + " " + lastName; }

    public static void printDetails()
    {   System.out.println("This is Employee " + lastName); }

    public Employee generateSon()
    { return new Employee(firstName+" Jr",lastName,salary/2); }
}
```

- a) **Write a constructor for class Employee that takes three parameters: a String for the firstName, a String for the lastName, and a double for the salary.**

Suppose that the following instructions are executed in another test class:

```
Person p1 = new Person("Mohamed","Ali");
Person p2 = new Person("Mohamed","Ali");
Employee e1 = new Employee("Abdallah","Said",1000);
p1 = e1;
```

- b) **What is the output of the instruction:**

```
System.out.println(p1);
```

**c) What is the output of the instruction:**

```
System.out.println(p1.getSalary());
```

**d) What is the output of the instructions:**

```
p1.printDetails();  
e1.printDetails();
```

**e) Explain the difference between method overloading and method overwriting:**

**f) Does the method generateSon() in Employee overloads or overwrites method generateSon() in Person class? (Justify your answer)**

**g) Is there a difference between the instructions:**

- p1 = (Person)e1;    and
- p1 = e1;

**h) Is there a difference between the instructions:**

- e1 = (Employee) p2; and
- e1 = p2;

**i) Which of f) and g) is performing down casting?**

**j) Write a static method printEmployees which takes as parameter an array of type Person and prints only the Employees objects inside that array.**



**Question 4.1: (10 Points)**

Given the definition of the following classes, print the output produced by the main method in the Test class.

```
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");
    }
}
```

Program Output

**Question 4.2: [20 Points]**

Define a class named Money to create objects of type money as follows. The class implements two interfaces: Clonable and Printable. Define these two interfaces: Clonable has one method clone and Printable has one method toString. Money class has three instance fields: sign, whole part, and hellah part. It has an internal class named Currency to define currency and exchange rate to US\$.

Define two constructors, clone, toString, and equals methods only