King Fahd University of Petroleum & Minerals

College of Computer Science and Engineering

Information and Computer Science Department

Second Semester 092 (2009/2010)

ICS 201 – Introduction to Computing II

Major Exam 1

Thursday, 25th March, 2010

Time: 120 minutes

Name: **Key Solution**

ID#:

Please circle your section number below:

Section	01	02	03	04
Instructor	Sami	Tarek	Sukairi	Sukairi
Day and Time	SMW 7 - 7:50	SMW 8 -8:50	SMW 9 - 9:50	SMW 13:00 – 13:50

Question #	Maximum Mark	Obtained Mark
1	15	
2	15	
3	30	
4	25	
5	15	
Total	**100**	

Question 1 [15 Points (5 + 10)]

a) What is the right answer?

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) Java does not use late binding for methods marked as:
   (a) final
   (b) static
   (c) private
   (d) 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:
   (a) concrete class
   (b) abstract class
   (c) discrete class
   (d) friendly class

5) If a method throws an exception, and the exception is not caught inside the method, then the method invocation:
   (a) terminates
   (b) transfers control to the catch block
   (c) transfers control to the exception handler

b) Given the following Shoe, TennisShoe, and WhiteTennisShoe classes:

```
class Shoe {
        public Shoe() {
                this("This is a shoe");
                System.out.println("Base Class");
        }
        public Shoe(String s)      {
                System.out.println(s);
}
}

class TennisShoe extends Shoe {
        public TennisShoe(){
                this("This is a Tennis Shoe");
                System.out.println("Derived Class");
        }
        public TennisShoe(String s) {
                super("Exam 1");
                System.out.println(s);
}
}

class WhiteTennisShoe extends TennisShoe {
        public WhiteTennisShoe(String s) {
                System.out.println(s);
}
}
```

What is the output of the following Test class?

```
class Test {
        public static void main(String args[]) {
                new WhiteTennisShoe ("A white tennis shoe is created");
        }
}
```

Exam 1
This is a Tennis Show
Derived Class
A white tennis shoe is created

Question 2 [15 Points]

a. Is multiple inheritance allowed for classes in Java? If yes, give an example. If no, explain why? (Multiple inheritance is when a class inherits from two base classes).

No, Java allows only one base class to be inherited because it eliminates inconsistent definitions of a single method.

b. What are the responsibilities of a class that implements a specific interface?

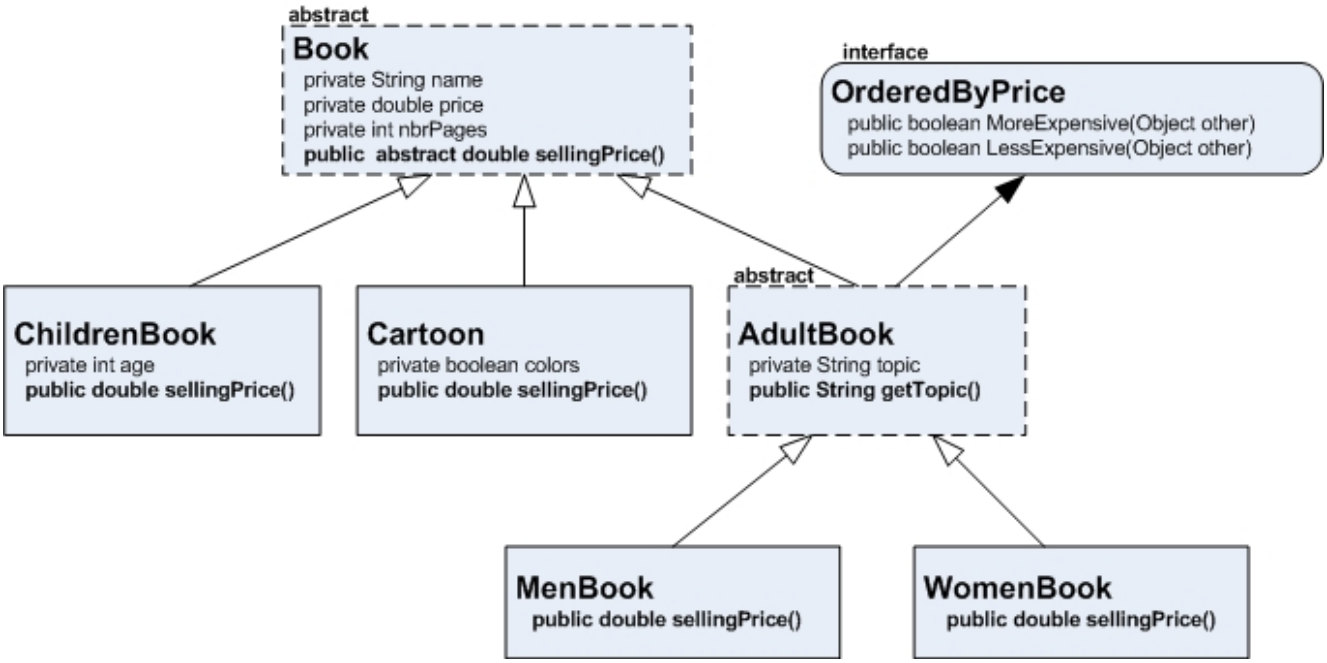
To implement an interface, a programmer must do two things. First, the phrase implements interface_name must be included at the start of the class definition. To implement more than one interface, the interface names must be separated by commas. The programmer must then implement all the methods listed in the definition of the interface.

c. What are the two advantages to using inner classes?

There are two big advantages to using inner classes. First, because they are defined within a class, they can be used to make the outer class self-contained. The second advantage is that the inner and outer classes' methods have access to each other's private methods and private instance variables.

Question 3 [30 Points]

The figure illustrates a Book inheritance hierarchy consisting of 6 classes and 1 interface. This illustration does not show all methods of the classes. It shows only the members that you need in this exercise. Don't worry about the remaining members.



The empty arrows designate an inheritance relationship (for example, ChildrenBook is derived from Book) and the filled arrow designate interface implementation (AdultBook implements OrderedByPrice interface). Abstract classes are clearly indicated.

The Book class is the base class, it is abstract and contains three instance variables: name, price, and nbrPages as well as one abstract method : sellingPrice().
ChildrenBook class has also an instance variable age and it overrides sellingPrice() method which returns the sellingPrice of the book. Cartoon class has colors instance variable and also overrides sellingPrice() method. Class AdultBook is also derived from Book, is abstract and implements OrderedByPrice interface. It has an instance variable topic and an accessor method getTopic(). Classes MenBook and WomenBook are derived from AdultBook but they are not abstract.

<u>Part I</u> [15 Points]

a) How many concrete classes are they in the above hierarchy? List them.

4 concrete Classes: ChildrenBook, Cartoon, MenBook, and WomenBook

b) Based on the figure above, write the heading (the first line) of class AdultBook.

```
public abstract class AdultBook extends Book implements OrderedByPrice
```

c) Implement the method MoreExpensive (interface OrderedByPrice) in class MenBook knowing that a MenBook object is more expensive than another MenBook object if its sellingPrice (not price) is larger or equal than the sellingPrice of the other MenBook object.

```
public boolean moreExpensiveThan(Object other)
{
   If(other == null)
        return false;
   else if (getClass() != other.getClass())
         return false;
   else
     {   MenBook mbobject = (MenBook) other;
         return (sellingPrice() >= mbobject.sellingPrice());
      }
```

Part II [15 Points]

Consider the following test class with a main method:

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

     Book bookList = new Book[6];

     bookList[0] = new ChildrenBook(…);
     bookList[1] = new MenBook(…);
     bookList[2] = new Cartoon(…);
     bookList[3] = new WomenBook(…);
     bookList[4] = new WomenBook(…);
     bookList[5] = new MenBook(…);

   }
}
```

a) Given the bookList array, write a code fragment that computes and prints the total selling price of all books in the array.

```
double sum = 0.0;
for(int i = 0; i < bookList.length; i++)
{
   sum += bookList[i].sellingPrice();
}
```

b) Given the same bookList array, write a code fragment that prints for every object of type AdultBook its topic.

```
for(int i = 0; i < bookList.length; i++)
{
     If(bookList[i] instanceof AdultBook)
      { AdultBook abobject = (AdultBook) bookList[i];
        System.out.println(abobject.getTopic());
      }
}
```

Question 4 [25 Points]

Write an abstract super class called Part, with two attributes: the partNumber, and a budgetCost for it. Write also three proper constructors. This class has two concrete subclasses: SelfManufacturedPart, and PurchasedPart. A self-manufactured part has a cost and a drawingNumber; it has also three constructors and a method returning whether it is over budget or under budget. A self manufactured part is over budget if its cost is larger than its budgetCost. A PurchasedPart has a set of suppliers, each with a price for the part. It also has a method to retrieve the lowest-cost supplier for a part and the corresponding cost, and also three proper constructors.

Note: write only the required methods. A supplier can be implemented as an inner class with two attributes: a name and a price.

```
public abstract class Part
{
    private String partNumber;  // or int
    private double budgetCost;

    public Part()
    {
            partNumber = "";
            budgetCost = 0;
    }

    public Part(String partNumber, double budgetCost)
    {
            this.partNumber = partNumber;
            this.budgetCost = budgetCost;
    }

    public Part(Part other)
    {
            this(other.partNumber, other.budgetCost);
    }

    public String getPartNumber()
    {
            return partNumber;
    }

    public double getBudgetCost()
    {
            return budgetCost;
    }
}
```

```
public class SelfManufacturedPart extends Part
{
	private double cost;
	private String drawingNumber;

	public SelfManufacturedPart()
	{
		super();
		cost = 0;
		drawingNumber = "";
	}

	public SelfManufacturedPart(String partNumber, double budgetCost, double cost,
String drawingNumber)
	{
		super(partNumber, budgetCost);
		this.cost = cost;
		this.drawingNumber = drawingNumber;
	}

	public SelfManufacturedPart(SelfManufacturedPart other)
	{
		this(other.getPartNumber(),other.getBudgetCost(), other.cost,
other.drawingNumber);
	}

	public boolean overBudget()
	{
		return (cost > getBudgetCost());
	}
}
```

```
public class PurchasedPart extends Part
{
	private Supplier[] sup;

	private class Supplier
	{
		private String name;
		private double price;

		public String getName()
		{
			return name;
		}

		public double price()
		{
			return price;
		}
	}

	public PurchasedPart()
	{
		super();
		sup = null;
	}

	public PurchasedPart(String partNumber, double budgetCost, Supplier[] sup)
	{
		super(partNumber, budgetCost);
		this.sup = sup;
	}

	public PurchasedPart(PurchasedPart other)
	{
		this(other.getPartNumber(),other.getBudgetCost(), other.sup);
	}

	public Supplier lowestCost()
	{
		Supplier lowest = sup[0];
		for(int i = 1; i < sup.length; ++i)
		if(sup[i].price < lowest.price) lowest = sup[i];
	    return lowest;
	}
}
```

Question 5 [15 Points]

a) What is the difference between Exception and Error in java? (2 Points)

- Exception and Error are the subclasses of the Throwable class.
- Exception class is used for exceptional conditions that user program should catch.
- Error defines exceptions that are not expected to be caught by the user program.

b) Differentiate between Checked Exceptions and Unchecked Exceptions? (3 Points)

- Checked Exceptions are those exceptions which should be explicitly handled by the calling method. Happen when programs interact with the world.Unhandled checked exceptions results in compilation error.
- Unchecked Exceptions are those which occur at runtime and need not be explicitly handled. RuntimeException and its subclasses, Error and it's subclasses fall under unchecked exceptions.

c) Is there anything wrong with this exception handling as written? Will this code compile? (4 Points)

```
try {
    //some correct code with throw instructions
} catch (Exception e) {
   //some correct code
 } catch (ArithmeticException a) {
    //some correct code
 }
```

This first handler catches exceptions of type Exception; therefore, it catches any exception, including ArithmeticException. The second handler could never be reached. This code will not compile.

d) Identify and name the potential sources of exception in the following statements. (6 Points)

```
int [ ] arr = null;
arr[0] = 1; //  NullPointerException
arr = new int [4];
int i, j;
```

```
            for (i = 0; i <= 4; i++)
                arr[i] = i; //  ArrayIndexOutOfBoundsException when i=4
            arr[i-1] = arr[i-1] / j; //ArithmaticException (Division by zero)
        PrintWriter out = new PrintWriter(new FileWriter("OutFile.txt")); //  IOException
```