**Philadelphia University**

****

**Faculty of Information Technology**

**Department of Software Engineering**

**Examination Paper**

**Lecturer : Ms. Enas Al-Naffar**

**Coordinator : Dr. Samer Hanna**

**Internal Examiner: Dr. Ali Fouad**

**Object-Oriented Programming 721220 Final Exam 2nd semester** **2012-2013**

**Date: 3rd June 2013**  **Section: 1**  **Time**: **120 Minutes**

**Information for Candidates**

1. *This examination paper contains 4 questions. The total is 40.*
2. *The marks for parts of questions are shown in round brackets.*

**I. Basic Notions**

*Objectives: The aim of the question is to evaluate your knowledge and skills concerning with the basic concepts of* ***OOP****.*

**Question 1: [`10 Marks]**

A- Choose the correct answer: [2 Marks, 1 Mark each]

1.  When an object has many forms, it has \_\_\_\_\_.

A)           Inheritance

B)           Scalability

C)           Encapsulation

D)           Polymorphism

2. What part of object-oriented technology defines super-class and sub-class relationships?

A)           Inheritance

B)           Scalability

C)           Encapsulation

D)           Polymorphism

B- Fill in the blanks with the correct answer: [4 Marks, 1 Mark each]

1. A variable known only within the method in which it is declared is called a(n) local variable.
2. Classes from which objects can be instantiated are called concrete classes.
3. It is possible to have several methods with the same name that each operate on different types or numbers of arguments. This feature is called method overloading.
4. Methods in a class that do not provide implementations must be declared using keyword abstract.

C- State whether each of the following is true or false. If a statement is false, explain why. [4 Marks, 1 Mark each]

1. Base class constructors are not inherited by derived classes. False; they are inherited
2. A has-a relationship is implemented via *inheritance.* False; Is-A not Has-A
3. A Car class has *is-a* relationships with the Steering\_Wheel and Brakes classes. False; Has-A not Is-A
4. When a derived class redefines a base class method by using the same signature and return type, the derived class method is said to overload that base class method. False; it's said to override.

### II. Familiar Problems Solving

### *Objectives: The aim of the question is to evaluate your basic knowledge of the key aspects of the lectures material and your ability to solve familiar problems.*

**Question 2:** **[15 Marks]**

Study the following class diagram, then wrote its corresponding c# code.



using System;

using System.Collections.Generic;

using System.Text;

public class location

{

 public double x, y;

}

abstract class shape

{

 protected location c=new location();

 public override string ToString()

 {

 return string.Format("{0}, {1}", c.x,c.y );

 }

 public abstract double area();

 public abstract double perimeter();

}

class circle : shape

{

 private double raduis;

 public override string ToString()

 {

 return string.Format("{0},{1}",base.ToString(),raduis);

 }

 public override double area()

 {

 return Math.PI \* raduis \* raduis;

 }

 public override double perimeter()

 {

 return Math.PI \* 2 \* raduis;

 }

}

class rectangle : shape

{

 private double side1,side2;

 public override string ToString()

 {

 return string.Format("{0},{1},{2}",base.ToString(), side1,side2);

 }

 public override double area()

 {

 return side1 \* side2;

 }

 public override double perimeter()

 {

 return 2 \* (side1 + side2);

 }

}

class TestShape

 {

 static void Main(string[] args)

 {

 rectangle r = new rectangle();

 circle c = new circle();

 Console.WriteLine( c.ToString());

 }

 }

**Question 3: [8 Marks]**

Study the following class, and then answer the questions below:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

 class Animal

 {

 protected double weight;

 protected double height;

 protected int x\_coordiantion;

 protected int y\_coordination;

 public animal(double h, double w)

 {

 x\_coordiantion=0;

 y\_coordination=0;

 weight=w;

 height=h;

 }

 public abstract void talk()

 {

 Console.WriteLine("print from the super class");

 }

 public void walk(double x, double y)

 {

 x\_coordiantion=x;

 y\_coordianion =y;

 }

 }

class cat:Animal

{

 private string name;

 public cat(double h, double w) : base()

 {

 name="Putchi";

 }

 public void talk()

 {

 return stirng.format("{0}", "meaw");

 }

 public void drink\_milk()

 {

 Console.WriteLine("iam drinking milk");

 }

}

class test

{

 Animal a = new cat();

 cat c = new cat();

 c.talk();

 c.walk();

 a.talk();

 a.walk();

 a.drink\_milk();

}

The three classes above have some errors . find these erorrs and correct them. **[8 marks]**

(at least four errors)

1- public abstract void talk() // should be virtual

 {

 Console.WriteLine("print from the super class");

 }

2- public cat(double h, double w) : base()// should be :base(h,w)

 {

 name="Putchi";

 }

3- public void talk()// should be public override void talk

 {

 return stirng.format("{0}", "meaw"); // the type is void so replace it with console.writeline

 }

4- class test // Main method is missing static void Main( ) { …. }

{

 Animal a = new cat();

 cat c = new cat();

 c.talk();

 c.walk();

 a.talk();

 a.walk();

 a.drink\_milk();

}

***III. Unfamiliar Problems Solving***

Objectives: *The aim of the question is to evaluate your knowledge of the key aspects of the lectures material and your ability to solve unfamiliar problems.*

**Question 4**: **[7 Marks]**

Extend the class **TestShape** in Question 2, by defining a List of type Shape.

* + Create a Method within the same class (**TestShape**) that adds a Shape into the List.
	+ Create a Method within the same class (**TestShape**) that deletes a Shape in a given location ( index)
	+ Use these newly defend methods in the Main method

class TestShape

 {

 static void Main(string[] args)

 {

 List<shape> P = new List<shape>();

 rectangle r = new rectangle();

 circle c = new circle();

 Console.WriteLine( c.ToString());

 Add(P,r);

 Delete(P,0);

 }

 public static void Add(List<shape> l,shape e)

 {

 l.Add(e);

 }

 public static void Delete(List<shape> l, int index)

 {

 l.RemoveAt(index);

 }

 }

Good Luck ☺