**Object-Oriented Application Development**

**Practical 3**

**Part A**

1. Consider the following program that consists of Employee class and EmployeeTest class.

The Employee class has:

* **Instance variables** representing name, hours worked, and pay rate
* **Properties** for all the instance variables with *get* members only
* **Constructor** with 3 parameters to initialise all the instance variables
* **Instance method** named ComputePay() to compute and return the pay

|  |
| --- |
| public class Employee  {  // instance variables  private string name;  private float hoursWorked;  private decimal payRate;  // properties with get member only  public string Name  {  get  {  return name;  }  }  public float HoursWorked  {  get  {  return hoursWorked;  }  }  public decimal PayRate  {  get  {  return payRate;  }  }  // constructor with 3 parameters  public Employee(string theName, float theHoursWorked, decimal thePayRate)  {  name = theName;  hoursWorked = theHoursWorked;  payRate = thePayRate;  }  // instance method  public decimal ComputePay()  {  return (decimal)hoursWorked \* payRate;  }  } |

The EmployeeTest class has a Main() method that does the following:

* Create 2 Employee objects
* Call the instance method ComputePay() for 2 Employee objects
* Display pay of the employees

|  |
| --- |
| using System;  public class EmployeeTest  {  public static void Main()  {  // create 2 Employee objects  Employee emp1 = new Employee("Jane Tan", 45, 72.0M);  Employee emp2 = new Employee("John Lee", 40, 50.0M);  // call instance method ComputePay() for the 2 Employee objects  decimal pay1 = emp1.ComputePay();  decimal pay2 = emp2.ComputePay();  // display name and pay of the employees  Console.WriteLine("Pay for {0} is {1}", emp1.Name, pay1);  Console.WriteLine("Pay for {0} is {1}", emp2.Name, pay2);  }  } |

Run the application and note the output.

1. Consider the following program that consists of a Student class and a StudentTest class:

The Student class has:

* **Instance variables** representing name and test scores for 2 tests
* **Properties** for all the instance variables with *get* members only
* **Constructor** with 3 parameters to initialise all the instance variables
* **Instance method** named ComputeAverage() to compute and return the average test score

|  |
| --- |
| public class Student  {  // instance variables  private string name;  private double score1;  private double score2;  // properties with get member only  public string Name  {  get { return name; }  }    public double Score1  {  get { return score1; }  }  public double Score2  {  get { return score2; }  }  // constructor with 3 parameters  public Student(string aName, double aScore1, double aScore2)  {  name = aName;  score1 = aScore1;  score2 = aScore2;  }  // instance method  public double ComputeAverage()  {  // local variable  double average;  average = (score1 + score2) / 2.0;  // note: call static method Ceiling() of Math class  return Math.Ceiling(average);  }  } |

The StudentTest class has a Main() method that does the following:

* Create a Student object
* Call the instance method ComputeAverage() for the object and display the result

|  |
| --- |
| using System;  public class StudentTest  {  public static void Main()  {  // create Student object  Student aStudent = new Student("Jane", 60.0, 66.0);  // display name and average of the student  Console.WriteLine("Average score for {0} is {1}",  aStudent.Name, aStudent.ComputeAverage());  }  } |

Average score for Jane is 63

Run the application and note the output.

1. Consider the following program that uses the Student class in Question 2 above.

|  |
| --- |
| using System;  public class StudentTest2  {  public static void Main()  {  Student aStudent = new Student("Jane", 80.0, 90.0);  double average = aStudent.ComputeAverage(80.0, 90.0);  //double average = aStudent.ComputeAverage();  Console.WriteLine("Average score: {0}", average);  }  } |

The code contains 1 error. Correct the error and run the application.

1. Consider the following program that consists of a Person class and a PersonTest class:

The Person class has:

* **Instance variables** representing name and identification number.
* **Properties** for all the instance variables with *get* members only
* **Three different constructor (constructor overloading)**

|  |
| --- |
| public class Person  {  // instance variables  private string name;  private string id;  // properties  public string Name  {  get { return name; }  }  public string Id  {  get { return id; }  }  // 1st constructor  // a default constructor - with no parameters and empty body  public Person()  {  }  // 2nd constructor with 1 parameter  public Person(string aName)  {  name = aName;  }  // 3rd constructor with 2 parameters  public Person(string aName, string anId)  {  name = aName;  id = anId;  }  } |

The PersonTest class has a Main() method that does the following:

* Create 3 Person objects using 3 different constructors

|  |
| --- |
| using System;  public class PersonTest  {  public static void Main()  {  // uses 1st constructor  Person person1 = new Person();  Console.WriteLine("Name is: '{0}'", person1.Name);  Console.WriteLine("ID is: '{0}'", person1.Id);  // uses 2nd constructor  Person person2 = new Person("Lillian");  Console.WriteLine("Name is: '{0}'", person2.Name);  Console.WriteLine("ID is: '{0}'", person2.Id);  // uses 3rd constructor  Person person3 = new Person("Julian", “123”);  Console.WriteLine("Name is: '{0}'", person3.Name);  Console.WriteLine("ID is: '{0}'", person3.Id);  }  } |

Run the application and note the output.

1. Consider the following class definitions which contain some errors. Identify and correct them.

|  |
| --- |
| public class NumberAdder() //should remove bracket  {  private int number1;  private int number2;  public void NumberAdder(int n1, int n2) // what is the error here? Remove void  {  number1 = n1;  number2 = n2;  }  public ComputeSum() // public int ComputeSum  {  return n1 + n2; // return number1 + number2;  }  } |

|  |
| --- |
| using System;  public class NumberAdderTest  {  public static void Main()  {  NumberAdder adder = new NumberAdder(); // NumberAdder(3,5)  int sum = ComputeSum(); // adder.ComputeSum();  Console.WriteLine("Sum is {0}", sum);  }  } |

using System;

public class NumberAdder // what is the error here?

{

private int number1;

private int number2;

public NumberAdder(int n1, int n2) // what is the error here?

{

number1 = n1;

number2 = n2;

}

public int ComputeSum() // what is the error here?

{

return number1 + number2; // what is the error here?

}

}

public class NumberAdderTest

{

public static void Main()

{

NumberAdder adder = new NumberAdder(3,5); // what is the error here?

int sum = adder.ComputeSum(); // what is the error here?

Console.WriteLine("Sum is {0}", sum);

}

}

**Part B**

1. Create a class named NumberCruncher that has the following:

* Instance variable that represents a single number (type *int*)
* Property for the instance variable with a get member only
* Constructor with one parameter to initializes the instance variable
* Instance method named ComputeSquare() that computes and returns the square of the number
* Instance method named ComputeCube() that computes and returns the cube of the number

using System;

public class NumberCruncher

{

// instance variables

private int number;

// properties with get member only

public int Number

{

get

{

return number;

}

}

// constructor with 1 parameters

public NumberCruncher(int theNumber)

{

number = theNumber;

}

// instance method

public int ComputeSquare()

{

return (int)number \* number;

}

public int ComputeCube()

{

return (int)number \* number \* number;

}

public class NumberCruncherTest

{

public static void Main()

{

NumberCruncher num = new NumberCruncher(3);

int square = num.ComputeSquare();

int cube = num.ComputeCube();

Console.WriteLine("Square and Cube = {0}, {1}", square, cube);

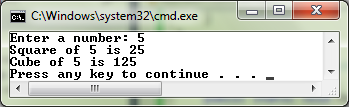
}

}

}

2. Create a class named NumberCruncherTest to test the NumberCruncher class in Question1. Allow the user to specify the number for the object.

Sample input and output



public class NumberCruncherTest

{

public static void Main()

{

string input;

int num1;

Console.WriteLine("Enter a Number:");

input = Console.ReadLine();

num1 = Convert.ToInt32(input);

NumberCruncher num = new NumberCruncher(num1);

int square = num.ComputeSquare();

int cube = num.ComputeCube();

Console.WriteLine("Square and Cube = {0}, {1}", square, cube);

}

}

}

3. Create a class named Circle to represent circles. A Circle object must have a radius and must be able to compute its area and circumference. The formula for area and circumference of a circle are:

area = π r 2

circumference = 2 π r

where r is the radius of a circle. Use Math.PI (a constant defined in the Math class) for π. Include a constructor and a property in your class.

4. Create a class named CircleTest to test the Circle class in Question 3. Allow the user to specify the radius for the object.

using System;

public class Circle

{

// instance variables

private double radius;

public const double PI = 3.14;

// properties with get member only

public double Radius

{

get

{

return radius;

}

}

// constructor with 1 parameters

public Circle(double theRadius)

{

radius = theRadius;

}

// instance method

public double ComputeArea()

{

return (double)PI \* radius;

}

public double ComputeCircum()

{

return (double)PI \* 2 \* radius;

}

public class CircleTest

{

public static void Main()

{

string input;

int num1;

Console.WriteLine("Enter a Number:");

input = Console.ReadLine();

num1 = Convert.ToInt32(input);

Circle num = new Circle(num1);

double area = num.ComputeArea();

double circ = num.ComputeCircum();

Console.WriteLine("Area and Circumference = {0}, {1}", area, circ);

}

}

}

1. Create a class named Item. Its data members are item number, description, unit price, and quantity purchased. DO NOT include a constructor. Include appropriate properties and a method that calculates the cost using the quantity and unit price.
2. Create a class name ItemTest to test the Item class in Question 5. Hardcode the values for the object, i.e. no need to get user input.

using System;

public class Item

{

// instance variables

public const double no = 999;

public const string desc = "Chocolate";

public const double price = 2.4;

public const double bought = 3;

// properties with get member only

public double No

{

get

{

return no;

}

}

public double Price

{

get

{

return price;

}

}

public double Bought

{

get

{

return bought;

}

}

// instance method

public double ComputeCost()

{

return (double)bought \* price;

}

public class ItemTest

{

public static void Main()

{

Item find = new Item();

double cost = find.ComputeCost();

Console.WriteLine("Item Number = {0} ", no);

Console.WriteLine("Item Description: {0}", desc);

Console.WriteLine("Price of Product: RM {0}", price);

Console.WriteLine("Price of 3 Choclate is = {0}", cost);

}

}

}

**Part C (Review)**

1. What are differences between instance variables and local variables?
2. What are differences between instance variables and properties?
3. What are differences between static methods and instance methods?
4. What is the purpose of a constructor?
5. What are overloaded constructors?
6. What is a default constructor? When do you automatically get a default constructor?

**Part D (Java)**

1. Write the Java programs for the questions in Part B. *Note*: Java does not support concept of properties. You need to create *setXxx()* and *getXxx()* methods that perform the actions of the set and get members of C# properties.