**Object-Oriented Application Development**

**Practical 8**

**Part A**

1. Consider the class definitions given below:

|  |  |
| --- | --- |
| public class ClassA  {  protected int a;  public int getA()  {  return a;  }  } | public class ClassB : ClassA  {  public void increaseA()  {  a++;  }  } |

1. Which class is the derived/sub/child class?

Class B

1. Which class is the base/super/parent class?

Class A

1. Will there be any error if access modifier protected is changed to private for instance variable a?

Yes. A++ will be affected

1. The classes above do not have constructors. The compiler will add default constructors for the classes. How will the default constructors look like?

public classA()

{}

public classB()

{}

2. Consider the Main() method given below which uses ClassB defined in Question 1:

|  |
| --- |
| using System;  public class TestClass  {  public static void Main()  {  ClassB obj = new ClassB();  obj.increaseA();  obj.increaseA();  Console.WriteLine(obj.getA());  }  } |

1. obj refers to an object of ClassB but the getA() method is defined in ClassA. Explain the object-oriented programming concept that makes the following valid: obj.getA().

The concept of inheritance that allow the sub class to inherit the…..

Yes due to the subclass inherit the method by class b.

1. What is the output of the program?

2

**Part B**

Consider the class definition for a Student class given below:

|  |
| --- |
| public class Student  {  private string name;  public string Name  {  get { return name; }  }  private int testMarks;  public int TestMarks  {  get { return testMarks; }  set { testMarks = value; }  }  public Student(string name)  {  this.name = name;  }  } |

1. Create an Undergraduate class as a subclass of Student class. An *undergraduate* student has a *name*, *test marks* and a *level* (an integer with values 1, 2, or 3). Include a constructor.

Test your class with the following test class:

|  |
| --- |
| using System;  public class StudentTest1  {  static void Main()  {  Student student = new Student("Tom");  student.TestMarks = 78;  Undergraduate undergraduate = new Undergraduate("Sam", 2); //level=2  undergraduate.TestMarks = 89;  Console.WriteLine("{0} got {1} marks",  student.Name, student.TestMarks);  Console.WriteLine("{0} got {1} marks",  undergraduate.Name, undergraduate.TestMarks);  }  } |

public class Undergraduate : Student

{

private int level;

public int Level

{

get { return level; }

set { level = value; }

}

public Undergraduate(string name, int level) :base(name)

{

this.level = level;

}

}

1. (a) Add a method called IsPass() in the Student class that determines if the student has passed the test. The method returns true or false. The passing test marks is 50.

public bool isPass()

{

if (TestMarks >= 50)

return true;

else

return false;

}

(b) Test Undergraduate class with the following test class:

|  |
| --- |
| using System;  public class StudentTest2  {  static void Main()  {  Console.Write("How many Undergraduate Students? ");  int count = Convert.ToInt32(Console.ReadLine());  Undergraduate[] students = new Undergraduate[count];  for (int i = 0; i < count; i++)  {  Console.Write("Enter name: ");  string name = Console.ReadLine();  Console.Write("Enter level: ");  int level = Convert.ToInt32(Console.ReadLine());  students[i] = new Undergraduate(name, level);  }  foreach (Undergraduate undergraduate in students)  {  Console.Write("Enter test marks for {0}: ", undergraduate.Name);  int marks = Convert.ToInt32(Console.ReadLine());  undergraduate.TestMarks = marks;  if (undergraduate.IsPass())  Console.WriteLine("Pass");  else  Console.WriteLine("Fail");  }  }  } |

1. Refer to the Student class above.
2. Create a Postgraduate class as a subclass of class Student. Include a constructor and a method called IsPass() that overrides the method in the Student class. For a *postgraduate* student, a pass is given if the test marks are 60 and above.

*Note*: you need to make some change to the IsPass() method in the Student class to allow the method to be overridden.

public class Postgraduate : Student

{

public Postgraduate(string name) : base(name)

{

return;

}

public bool IsPass()

{

if (TestMarks >= 60)

return true;

else

return false;

}

}

1. Test Postgraduate class with the following test class:

|  |
| --- |
| using System;  public class StudentTest3  {  static void Main()  {  Postgraduate postgraduate = new Postgraduate("Tim");  Console.Write("Enter test marks for postgraduate student {0}: ",  postgraduate.Name);  int marks = Convert.ToInt32(Console.ReadLine());  postgraduate.TestMarks = marks;  if (postgraduate.IsPass())  Console.WriteLine("Pass");  else  Console.WriteLine("Fail");  }  } |

1. Refer to the Student, Undergraduate, and Postgraduate classes above.

(a) Add a ToString() method in the Student class that returns the *name* of the student. The string returned should be in the format:

Name: xxx

public class Student

{

public string name;

public string Name

{

get { return name; }

}

private int testMarks;

public int TestMarks

{

get { return testMarks; }

set { testMarks = value; }

}

public Student(string name)

{

this.name = name;

}

public override string ToString()

{

return String.Format("Name:{0}",

name);

}

}

public class Undergraduate : Student

{

private int level;

public int Level

{

get { return level; }

set { level = value; }

}

public Undergraduate(string name, int level) : base(name)

{

this.level = level;

}

public override string ToString()

{

return String.Format("Name:{0}",

name);

}

}

public class Postgraduate : Student

{

public Postgraduate(string name) : base(name)

{

return;

}

public bool IsPass()

{

if (TestMarks >= 60)

return true;

else

return false;

}

public override string ToString()

{

return String.Format("Name:{0}",

name);

}

}

(b) Add a ToString() method in the Undergraduate class that returns a string with the *name* and *level*. The method should call the ToString() method in the Student class. (*Note*: use keyword base.) The string returned should be in the format:

Name: xxx Level: x

public override string ToString()

{

return String.Format("Name:{0}, Level:{1}",

name, level);

}

1. Test the modified with the following test class:

|  |
| --- |
| using System;  public class StudentTest4  {  public static void Main()  {  Student student = new Student("Jim");  Undergraduate undergraduate = new Undergraduate("Tom", 2);  Postgraduate postgraduate = new Postgraduate("Tim");  Console.WriteLine("Student:");  Console.WriteLine(student.ToString());  Console.WriteLine("Undergraduate:");  Console.WriteLine(undergraduate.ToString());  Console.WriteLine("Postgraduate:");  Console.WriteLine(postgraduate.ToString());  }  } |