Rewarding Learning

ADVANCED
General Certificate of Education
2023

Software Systems Development

Unit AS 1

Introduction to Object Oriented Development

[SDV11]

FRIDAY 26 MAY, MORNING

MARK SCHEME

AVAILABLE MARKS

1

Object Oriented term	Object Oriented definition	Mark
Object	An object is an instance of a class. An object will contain all the attributes, properties and methods of the class it is created from. Many objects can be created from the same class.	One each of any two: [1] for explaining an object is an instance of a class [1] for explaining that an object will contain all the attributes, properties and methods of a class [1] for explaining the many methods can be made from one class
Constructor	A type of method within a class which is executed when an instance of the class is created.	[1] for constructor [1] parameterised constructor
Encapsulation	The process of defining a class by **concealing its internal members** from **outside the class** and accessing those internal data members only through **public methods or properties**.	[1] for explaining the use of concealment of class members [1] for explaining this is from outside of the class [1] for explaining how they can be accessed [1] A group of variables, properties and methods grouped together.
Virtual method	**A method with an implementation that can be overridden by a derived/ sub/child class**.	[1] for overridden in the correct context [1] Any of these derived/sub/child class. Used in the correct context

[1] each correct [8] 8

2 **(i)**

```
public Child(int Childid, string foreName, string surName, bool equipneeded,
double amountpaid, double spendingmoney)
    {
        ChildID = childID;
        Forename = forename;
        Surname = surname;
        EquipNeeded = equipNeeded;
        AmountPaid = amountPaid;
        SpendingMoney = spendingMoney;
        CostofTrip = 900;
    }
```

parameter types, [1]
all parameters [1]
no CostofTrip passed [1]
all correct assignment [1]
CostofTrip assignment set to 900 [1] [5]

(ii)
```
public int CostofTrip
{
    get { return costofTrip; }
    set { costofTrip = value; }
}
```

```
Java example
//getter
public int getCostofTrip()
{
        return costofTrip;
}
    //Setter method
    public void setCostofTripint value) {
        costofTrip; = value;
}
```

Return Type int [1]
No brackets [1]
get [1]
set [1] [4]

(iii)
```
public int TripCost()
    {
      if(!equipNeeded)
      {
          return costofTrip;
      }
      else
      {
          return costofTrip + 200;
      }
    }
```

Return Type int [1]
Correct header including brackets [1]
Check for equipment [1]
Correct return [1] [4]

(iv)
```
public bool validSpendingMoney()
      {
          if (spendingMoney<=400)
          {
              return true;
          }
          else
          {
              return false;
          }
      }
```

Access modifier and return Type bool [1]
No parameters [1]
Check of spendingMoney variable [1]
Correct return [1] [4]

(v) Child objChild = new Child(int ChildID,string forename, string surname,equipNeeded, double amountPaid, double spendingMoney, 900);

Reference to Child class [1]
Assigning new object a name [1]
Correct placement of the new command [1]
parameters getting passed [1]

```
Child objChild =
new Child (
1, "John", "Smith",
true, 300, 400
900 )
```

[4]

(vi) Child [ ] childArray = new Child[60];

Explain the line of code with reference to **four** of the following:

References objects of type Child [1]
childArray set as array name [1]
Using the new command to create the array [1]
Instantiate array size 60 [1]
initialised to null [1]
Creates space in the memory [1]

[4]

AVAILABLE MARKS

25

			AVAILABLE MARKS
3	**(a)**	IOException SerializableException IndexOutOfRange FileNotFound Exception FormatException DividebyzeroException SystemException [1] each for any two [2]	

	(b)	public void convertToint(string num)
		{
		try
		{
		Console.WriteLine(int.Parse(num));
		
		}
		catch(Exception ex)
		{
		Console.WriteLine("The value could not be converted to an integer");
		}
		
		}
		
		Return type void [1]
		Correct use of try [1]
		Attempt to convert string [1]
		Correct use of catch [1]
		Correct output message [1] [5]
			7

4 (a)

```
        bool flag = true;
        string suggestedName = forename.Substring(0,1) + surname
+number.ToString();

        while (flag)
        {
            for(int x=0;x< Usernames.Length;x++)
            {
                if (Usernames[x]==suggestedName)
                {

        number=random.Next(100,1000)
        suggestedName = forename.Substring(0, 1) + surname + number.
        ToString();
        break;
                }
                else
                {
                    flag = false;
                }
            }
        }

        return "The new employee username is: " + suggestedName;
    }
```

Declaration of variable to control the while/do while loop [1]
Get first letter of forename [1]
Concatenation of surname [1]
Concatenation of random number[1] and casting to string [1]
While Loop [1]
Loop of array with correct termination [1]
Check if username is equal to the value in array [1]
Generate a new username if not unique [1]
Get new random number [1]
Set flag to false if not found [1]
Return new username [1] [12]

(b)

```
public void Emails()
    {

        for (int x = 0; x < Usernames.Length; x++)
        {

        Console.WriteLine(Usernames[x]+ " " +
        Usernames[x] + "@ccea.co.uk");

        }

    }
```

Return type void [1]
Loop [1]
Creating new email [1]
Output [1] [4]

AVAILABLE MARKS

16

				AVAILABLE MARKS
5	**(a)**	**(i)**	Selection, Bubble, Quicksort, Insertion or other [1] for any of these description of sequence [1] pass [1] swap method [1] termination [1] [5]	
		(ii)	Sample illustration of bubble 40, 64, 18, 7, 99 1st pass 40,18,7,64,99 2 swaps(true) 2nd pass 18,7,40,64,99 2 swap (true) 3rd pass 7,18,40,64,99 1 swap (true) 4th pass 7,18,40,64,99 0 swap (false) – terminate 1 each for any two correct swaps [1] × 2 any correct pass list involving swap [1] correct termination [1] all correct [1] [5]	
	(b)	**(i)**	An interface is a **fully abstract class**, this means it can only contain abstract methods and properties. To use an interface method, the interface must be implemented using the : in the heading of a class after the class name. The body of the interface method is provided by the "implement" class. Any **three** from used in the correct context: Interface fully abstract – cannot have implementation [1] No access modifiers [1] No constructors [1] Simulates multiple inheritance [1] Class must implement the interface methods [1] Any other valid answer [3]	
		(ii)	An abstract class cannot be instantiated, it has to be used via inheritance. The purpose of an abstract class is to provide a common definition of a base class that multiple derived classes can share. Any **two** from below [1] for explaining it cannot be instantiated [1] for explaining it can be used via inheritance [1] explain that it is often used as a base class [1] can include both abstract and non-abstract methods and properties [2]	15

AVAILABLE MARKS

6 **(a)** Any **four** from below or any other suitable answer

- Inheritance allows for the inheriting of all properties from a base class
- A derived class can access all the functionality of an inherited class
- Polymorphism uses overriding
- It implements reusability of code
- testing required only for the differences within the derived class
- the process of overloading can be used
- single class inheritance is supported only
- multiple interface inheritance is allowed
- saves time for testing or development
- needs to be used in the correct context [4]

(b)

```
class Checkup:Booking
    {
        string issueDescription;
        DateTime length;
        double Checkupprice;


        public Checkup(int BookingID,int PetID,DateTime
        bookingDate,DateTime bookingTime, string issuedescription,
            DateTime length, Double
        checkupPrice):base(bookingID,PetID,bookingDate,bookingTime)
            {
                IssueDescription = issuedescription;
                Length = length;
                double CheckupPrice = checkupPrice;

            }
}
```

Checkup : **Booking** [1] or Checkup **extends Booking** [1]
fields definitions [1]
fields passed for Booking and Checkup [1]
:base [1]
Passing of Booking fields [1]
Assignment of Checkup fields only [1]
Any other suitable answer [6]

(c) **(i)**

```
public bool CheckDate(DateTime bookingDate)
    {
        if (bookingDate < DateTime.Today.AddDays(7))
        {
            return false;
        }
        else
        {
            return true;
        }

    }
```

Return type bool [1]
Data parameter passed [1]
If statement [1]
Comparison of date (must be 7 days) [1]
Correct return [1] [5]

(d) C#

```
public override double TotalPrice()
{
return base. TotalPrice()+ medPrice;
}
```

Java

```
public override double TotalPrice()
{
return super. TotalPrice()+ medPrice;
}
```

Use of key word override [1]
Return type double [1]
Call base method TotalPrice() [1]
Calculation of price [1]
Return [1] [5]

AVAILABLE MARKS
20

7 public void totalProfit(Checkup[] checkups)

```
        {
            double highestCost = 0.00;
            double profit = 0.00;
            int position = 0;

            for (int x=0;x<checkups.Length;x++)
            {
                profit + = checkups[x].totalPrice() * 0.15;

                if(checkups[x].totalPrice()> highestCost)
                {
                    highestCost = checkups[x].totalPrice();
                    position = x;
                }
            }
Console.WriteLine("The amount of profit generated from the weekly check-ups
was: " + profit.ToString("c"));

Console.WriteLine("Details of the highest priced check-up are:
"+checkups[position].IssueDescription +"\n"+ checkups[position].Length +"\n"+
checkups[position].CheckupPrice);


        }
```

Return type [1]
Passing the array in the parameters [1]
Declaring variables[1]
Use of loop with correct terminator [1]
Call of totalPrice() [1]
Correct running total calculation [1]
Checking if current output from totalPrice was highest value [1]
Output message returning the total profit [1]
Output message returning the highest checkup object details [1] [9]

AVAILABLE MARKS
9

Total | **100**