Rewarding Learning

ADVANCED
General Certificate of Education
2019

Software Systems Development

Unit AS 1

Introduction to Object Oriented Development

[SDV11]

WEDNESDAY 22 MAY, AFTERNOON

MARK SCHEME

AVAILABLE MARKS

1 Underline the most appropriate word from those given in the brackets that will correctly complete each statement.

An object is an (example, **instance**) of a class that can perform a set of related activities that define its (**behaviour**, operation).
Abstract classes (can, **cannot**) be instantiated.
An abstract class can only be used as a (**super/base class**, interface, derived/ subclass) for other classes that (change, **extend**) it.
A class can inherit from (**one**, many) abstract class(es) and must (overload, **override**) all the methods/properties that are declared to be (**abstract**, virtual) methods/properties.
Polymorphism at run-time is a primary concept of object-oriented programming which is enabled through (early binding, **late binding**).

[1] for each item in bold underlined [9] 9

2 **(a)** Methods are fundamental building blocks of programming languages. Give three advantages for their use.

Reuse of code
Structured design to simplify solution
Multiple developers
Development time improved
Simplified testing
Facility of overloading and overriding
or other relevant reason
([1] each for any three) [3]

(b) string [ ] Months = {"JAN", "FEB"......"DEC"};

Type	[1]
Array	[1]
Contents	[1]

[3]

(c) **(i)** Return the average income for the year;

JAVA solution

```
public static double averageIncomeI( int[][] MonthlyBookings)
{     double total = 0.00;
      for(int x =0; x< monthlybookings [0]. length; x++) // accept months
      total+= MonthlyBookings[x][0]* 2.5 + MonthlyBookings[x][1]* 1.5;
      return total / MonthlyBookings.length;
```

C# Solution

```
      public static double averageIncome( int[,] MonthlyBookings)
      {     double total = 0.00;
            for (int x = 0; x < MonthlyBookings.GetLength(0); x ++)
                total + = MonthlyBookings [x,0] * 2.5
                       + MonthlyBookings [x,1] * 1.5;
            return total/MonthlyBookings.Getlength(0);
      }
```

header return type	*[1] (allow omission of parameters)*
Declare/Initialise total	*[1]*
Loop	*[1]*
Running Total 2D index	*[1]*
Correct calculation (Adult price and child price)	*[1]*
Return average	*[1]*

[6]

AVAILABLE MARKS

(ii) Output the names of the months with income greater than or equal to the average income for the year;

```
JAVA
public static void  avgMonthNamesl( int[][] MonthlyBookings, string[]
Months)
{       double income, avg;
         avg = averageIncome(MonthlyBookings);
        for( int x =0; x< months.length;x++)
        {
                income = MonthlyBookings[x][0]* 2.5 + MonthlyBookings[x]
                [1]* 1.5;
                if( income >=avg)
                            System.Printline ("    ", Months [x])
        }

}       C# Solution
        public static void  avgMonthNames( int[,] MonthlyBookings, string[]
        Months)
        {       double income, avg;
                avg = averageIncome (MonthlyBookings);
                for (int i = 0; i< Months.Length; i++)
        {

                Income = MonthlyBookings [i,0] * 2.5 +
                            MonthlyBookings [i,1] * 1.5;
                if (income > = avg)
                        Console. WriteLine (Months [i]);
        }
}
```

return type	*[1]*
use of average	*[1]*
call method	*[1]*
Loop	*[1]*
comparison	*[1]*
output month name	*[1]*

[6] 18

AVAILABLE MARKS

3 **(a)** Exception [1]

(b) Description referencing the following appropriately
try
catch
finally
Surround relevant code for entry of date
Reference exception caught
error message
[1] each for any five [5]

(c) **(i)**

Header return type	*[1]*
Pass parameter	*[1]*
current date	*[1]*
add/subtract 18 years	*[1]*
comparison structure	*[1]*
correct comparison logic	*[1]*

[6]

C# sample answer

```
public Boolean ValidAdult(DateTime myDOB)
(
        DateTime chkDate = DateTime.Now.Date;
        chkDate = chkDate.Date.AddYears(-18);
        if(chkDate.CompareTo(myDOB.Date)  <0)
                return  false;
        return true;
)
```

Java   sample answer

```
public bool ValidDOB(Date  myDOB)
(
        Date  chkDate =new Date()

        Calendar c = Calendar.getInstance();
        c.setTime(chkDate);
        c.add(Calendar.YEAR, -18);
        chkDate = c.getTime()

        if(chkDate.CompareTo(myDOB.Date)  <0)
                return  false;
        return true;

        // or chkDate.before(myDate)            returns true / false
```

(ii)

```
try
{  ....  myApp. ApplicantDOB = new DateTime (int yearDOB, int
monthDOB, int dayDOB);
}
catch(DOBException dobEx)
{       // catches exception thrown from set method in Applicant class
        Error message using dobEx
}
catch(Exception ex)
{       // catches general exception for the illegal date
        Error message indicating non-date entry
}
```

			AVAILABLE MARKS
try	*[1]*		
surround date code	*[1]*		
catch customised exception	*[1]*		
output customised error	*[1]*		
catch general error (ordered	*[1]*		
output date format error	*[1]*	[6]	18

4 **(a)** overloading – reference to

same name, different signatures, return types may differ, (allow multiple methods)
[1] each for any 3
Any example showing name and signatures [1]

Interface – appropriate reference to three of the following:
abstract, methods /properties, facilitates multiple inheritance, imposes structure, no implementation code
any three
[1] each
any Interface e.g. IComparable [1] [8]

(b) **(i)** private Book [] arrBook = new Book[50];

Array	[1]	
Type	[1]	
New	[1]	
Size	[1]	[4]

(ii)

```
public double income(Book [] arrBook)
{
      double total = 0.0;
      for( int x=0; x<arrBook.Length; x++)
            total+=arrBook[x].Price * arrBook[x].NoSold;
      return total;
}
```

Visibility	*[1]*		
Type	*[1]*		
Parameter	*[1]*		
Declaration total	*[1]*		
Running total	*[1]*		
Calculation	*[1]*		
Index	*[1]*		
Return	*[1]*	[8]	20

AVAILABLE MARKS

5 **(a)** *Visibility – private* *[1]*
Reason – internal class method for clarity
/not relevant to user *[1]* [2]

(b) *Loop – correct terminator* *[1]*
Check for digits *[1]*
Check final character *[1]*
Return of (true and false) *[1]* [4]

C# Example

```
private boolean validBookFormat (string bookCode)
{
        for (int x=0; x<9; x++)
        {
                if(!bookCode[x].isDigit())
                        return false;
        }
        if(bookCode[9].isDigit() || bookCode[9] == 'x')
                        return true;
        return false;
}
```

(c) *Initialise total or weight* *[1]*
Correct character to numeric value *[1]*
Loop from 0–9 *[1]*
Correct multiplier *[1]*
Running total *[1]*
Division by 11 *[1]*
Convert x *[1]*
Correct return (true and false) *[1]* [8]

```
private boolean validBookRelationship (string bookCode)
{
        Int total =0, weight =10;
        for (int x=0; x<9; x++)
        {
                total+=bookCode[x] *weight;
                weight--;
        }
        if( bookCode[9] == 'X')
                total+=10;
        else
                total+=bookCode[x];

        if( total % 11 == 0 )
                return true;
        else
                return false;
}
```

Alternative solutions acceptable 14

					AVAILABLE MARKS
6	**(a)**	**(i)**	derived / sub / child	[1]	
		(ii)	overriding	[1]	
		(iii)	polymorphic	[1]	
	(b)		No	[1]	
			Session can be a single class – taster	[1]	5

7 **(a)**

```
:Session or extends Session            [1]
private   char        dailyWeekly;
private   int         frequency;
fields above [1] ([0] mark if all fields included)

public Course (int sessionNo, String description, char sLevel,
                int roomCapacity, DateTime dateTimeStart,
               char dailyWeekly, int frequency)
     :base ( sessionNo, description, sLevel,roomCapacity, dateTimestart)
{
     DailyWeekly = dailyWeekly;
     Frequency = frequency;
}
```

Session fields *[1]*
Course fields *[1]*
Base *[1]*
Pass of Session fields *[1]*
Assignment of Course fields *[1]* [7]

(b)

```
public override   double cost()
{
     return base.cost()  * frequency;  //C#
          super.cost()                 //java
}
```

override *[1]*
return type *[1]*
call base cost() *[1]*
calculation *[1]* [4]

(c)

```
public  double MaxProfit()
{
return cost() * roomCapacity – venueCharge
}
```

Header no parameters *[1]*
Return type *[1]*
return *[1]*
call cost() in Course *[1]*
calculation *[1]* [5] 16

Total **100**