Rewarding Learning

ADVANCED SUBSIDIARY (AS)
General Certificate of Education
2016

Software Systems Development

Unit AS 1:

Introduction to Object Oriented Development

[A1S11]

WEDNESDAY 25 MAY, AFTERNOON

MARK SCHEME

General Marking Instructions

Introduction

Mark schemes are published to assist teachers and students in their preparation for examinations. Through the mark schemes teachers and students will be able to see what examiners are looking for in response to questions and exactly where the marks have been awarded. The publishing of the mark schemes may help to show that examiners are not concerned about finding out what a student does not know but rather with rewarding students for what they do know.

The Purpose of Mark Schemes

Examination papers are set and revised by teams of examiners and revisers appointed by the Council. The teams of examiners and revisers include experienced teachers who are familiar with the level and standards expected of students in schools and colleges.

The job of the examiners is to set the questions and the mark schemes; and the job of the revisers is to review the questions and mark schemes commenting on a large range of issues about which they must be satisfied before the question papers and mark schemes are finalised.

The questions and the mark schemes are developed in association with each other so that the issues of differentiation and positive achievement can be addressed right from the start. Mark schemes, therefore, are regarded as part of an integral process which begins with the setting of questions and ends with the marking of the examination.

The main purpose of the mark scheme is to provide a uniform basis for the marking process so that all the markers are following exactly the same instructions and making the same judgements in so far as this is possible. Before marking begins a standardising meeting is held where all the markers are briefed using the mark scheme and samples of the students' work in the form of scripts. Consideration is also given at this stage to any comments on the operational papers received from teachers and their organisations. During this meeting, and up to and including the end of the marking, there is provision for amendments to be made to the mark scheme. What is published represents this final form of the mark scheme.

It is important to recognise that in some cases there may well be other correct responses which are equally acceptable to those published: the mark scheme can only cover those responses which emerged in the examination. There may also be instances where certain judgements may have to be left to the experience of the examiner, for example, where there is no absolute correct response – all teachers will be familiar with making such judgements.

			AVAILABLE MARKS
1	False	[1]	
	True	[1]	
	False	[1]	
	True	[1]	
	True	[1]	
	False	[1]	6

2 **(a)**

```
double discountRate = 0.0, saleAmount = 75.0;
char rating = 'A';
if (saleAmount >= 75.0 && rating != 'C' )
discountRate = 0.1; (allow 1)
```

[1] for initialization of discountRate at start or at else ...
[1] decision saleAmount [1] && [1] rating
[1] set discount discountRate 0.1 Allow 10 or 0.9
Allow !rating = 'C' [5]

(b) sample answer:

```
String sentence = "A cow jumped over the anvil and the fence";
String word;

if(sentence.Contains(" the "))                                          [1]
{
  sentence = sentence.Replace(" the ", " a ");                          [1]
  for (int x =0; x<sentence.Length -3; x++)   [1] correct loop [1] end index [2]
 {
    if( sentence.Substring(x,3)==" a ")       [1] check 'a'  [1] as a word  [2]
    {
       word =sentence.Substring(x+3, 2);

       switch(word[0])                                                  [1]
       {
          case 'a': //
          case 'e': //
          case 'i': //
          case 'o': //
          case 'u': sentence=sentence.Substring(0,x+2)  +'n'  + sentence.
Substring(x+2);  break;

       }   [1] for vowel check [1] for 'n'  [1] for correct index  [1]   [3]
    }
 }

 Console.Write("\n\n\tChanged sentence   "+ sentence);

}

else
   Console.Write("\n\n\tNo change to  "+ sentence);                      [1]
(or any relevant answer)
```

Any valid alternative
check " the " 1 mark
replace " the " 1 mark
check " a " or " the " 1 mark 1 mark as word
check the consonant 1 mark
end loop 1 mark

16

switch or loop for vowels 1 mark
check vowel 1 mark
add n or an 1 mark
correct index 1 mark
No change 1 mark

3 (a) Constructor – C# sample

```
public Booking(int bookingNo, String clientName, char package, DateTime
bookingDate, int noInParty)
{
    this.bookingNo = bookingNo;
    this.clientName = clientName;
    this.package = package;
    this.bookingDate = bookingDate;
    this.noInParty = noInParty;
}   ([1] parameters, [1] data type [1] any other correct
    assignment)                                                    [3]
```

GET / SET c# sample

```
public int NoInParty    type [1] capital letter [1]                [2]
{
    get { return noInParty; }                                      [1]
    set { noInParty = value; }                                     [1]
}
```

OR GET / SET java example

```
public int getNoInParty()   [1] alt
{
    return  noInParty;      [1] alt
}
public void setNoInParty (int noInParty)   { [1]alt
    this. noInParty = noInParty [1]alt
}
```

Method calcCost

```
     public double calcCost()                                      [2]
    ([1] return type, [1] name no parameters)
    {
       double price = 0.0;                                         [2]
    ([1] type, [1] initialisation)
    switch (package)                                               [1]
    {
          case 'A': price = 50.0; break;
          case 'B': price = 75.0; break;
          case 'C': price = 125.0; break;
          case 'D': price = 395.0; break;        ([1] any one correct )   [1]

}
if (noInParty >= 6 && (package == 'C' || package == 'D'))          [3]
([1] noInParty comparison [1] && brackets, [1]OR package)
```

					AVAILABLE MARKS
			return noInParty * price * 1.15;	[2]	
			([1] correct calculation [1] surcharge calculation)		
			else		
			return noInParty * price;	[2]	
			([1] return, [1] calculation)		
			}		
			public double calcDeposit()		
			{		
			return calcCost()/10;	[2]	
			([1] calculation [1] method call)		
			}		
			public double calcOutstandingCost()		
			{		
			return calcCost() - calcDeposit();	[2]	
			([1] calculation [1] method calls)		
			}		
	(b)	**(i)**	instantiation of an object of type Booking	[1]	
			method call for today's date from class DateTime	[1]	
		(ii)	Console.Write("\n\n\t Outstanding Cost is {0:.00} " , booking.calcOutstandingCost());	[1]	27
4	**(a)**	**(i)**	data type – alphabetic, numeric, alphanumeric; format – A999 letter followed by three digits, email address; range – number lying between 2.00 and 15.00; presence – surname, forename, address exists; compatibility – height/weight; dependancy – cost/selling price. ([1] each any three)	[3]	
		(ii)	examples noInParty data type/range clientName presence/length/data type ([1] any two appropriate checks)	[1]	
	(b)		Describe implementation of Exception Handling with reference to try catch finally many different classes available or inbuilt an example error messages		
		(i)	apply appropriate checks to value before SET of field An error causes a throw of an exception object. Customised class must extend Exception class. ([1] each for any six) Or any other valid	[6]	
		(ii)	try{ } surrounding input of value and set of value catch{ } deals with customised exception/Exception finally{ } tidy up code ([1] each aspect emboldened)	[3]	13

AVAILABLE MARKS

5 **(a)** Abstract class cannot be instantiated as accommodation is common data and must be extended to denote a specific type.
([1] non instantiation, [1] commonality, [1] extended) [3]

(b)

```
public double incomeYear()
{
    return rent * noRentPaymentsPerYear;
}
```

([1] return, [1] calculation) [2]

(c) **(i)**

```
class Flat:Housing          ([1] class Flat, [1] extends Housing)
{
    double maintenanceCharge;    ([1] field/[0] if others included)

    public Flat(int accommodationNo, String address1, String address2,
        String postcode, double valuation, double rent, int
        noRentPaymentsPerYear,
        String type, int noBedrooms, int noCarParkingSites,
        double maintenanceCharge)
            ([2] all fields/[1] maintenanceCharge only,)

        : base(accommodationNo, address1, address2,
        postcode, valuation, rent, noRentPaymentsPerYear, type,
        noBedrooms, noCarParkingSites)
            ([1] pass base, [1] fields to Housing:Accommodation)
    {
        this.maintenanceCharge = maintenanceCharge;    ([1])
    }
```

[9]

GET/SET [1] mark (Do not penalise Header again)

(ii)

```
public override double incomeYear() ([1] override, [1] no parameters)
{
    return base.incomeYear() + maintenanceCharge;
}              ([1] base call, [1] add maintenanceCharge)
```

[4]

(iii) override [1]

(d)

```
double incomeTotal = 0.0;                    ([1] initialisation)
for (int x = 0; x < accArray.Length; x++)    ([1] loop)
{
        incomeTotal += accArray[x].incomeYear();
}
        ([1] add, [1] method call)

Console.WriteLine("\n\tYearly Income {0:.00} ", incomeTotal);
                    ([1])
```

[5]

24

				AVAILABLE MARKS
6	**(a)**	: IComparable/implements Comparable – implements an interface – to allow comparison of two objects – return type is integer-0 if same, -ve if smaller,+ve if greater. – Allows use of Array/Arrays class – Sort an array of objects – applies structure – inherits methods – empty methods in interface – polymorphism – stimulates multiple inheritance [1] each for any four	[4]	

	(b)	public int CompareTo(Object obj)
		{
		Accommodation other = obj as Accommodation;
		return this.rent.CompareTo(other.rent);
		}
		([1] header, [1] – cast , [1] comparison, [1] return
		[1] – using call CompareTo)
			[5]	

	(c)	Array.Sort(accArray);
		Console.WriteLine("\n\tProperties in order of rent charged\n\n”);
		for (int x = 0; x < accArray.Length; x++)
		{
		Console.WriteLine(accArray[x].ToString());
		}
		(SORT – [1] class Array, [1] array parameter)
		([1] loop [1] output index, [1] array name)
		Alternative solution
		Allow sorting routine
		naming routine
		reference to rent field
		swap of object
		loops
		Any two, one mark each
			[5]	14

			Total	**100**