

1(a)(i)	DECLARE Member1 : ClubMember
1(a)(ii)	One mark for each correct answer Example answer <pre>Member1.Code ← 984632 Member1.FeesPaid ← TRUE</pre>
1(b)(i)	One mark per mark point (Max 2) <pre>MP1 TYPE Activity= MP2 (Badminton, Football, Golf, Snooker, Swimming, Tennis)</pre> Example answer <pre>TYPE Activity = (Badminton, Football, Golf, Snooker, Swimming, Tennis)</pre>
1(b)(ii)	DECLARE Choice : Activity

1(a)(i)	One mark per mark point (Max 2) <pre>MP1 TYPE Spectrum = MP2 (Red, Orange, Yellow, Green, Blue, Indigo, Violet)</pre> Example answer <pre>TYPE Spectrum = (Red, Orange, Yellow, Green, Blue, Indigo, Violet)</pre>
1(a)(ii)	One mark per mark point (Max 2) <pre>MP1 the list is ordered/ordinal MP2 the list contains all possible values MP3 no duplicate values in list MP4 all values are the same data type</pre>
1(b)	One mark for TYPE ColourData and ENDTYPE correct One mark for correct use of DECLARE in all declarations One mark for correct use of Spectrum in declaration One mark for remaining four declarations correct (STRING, INTEGER, REAL, BOOLEAN) Example answer <pre>TYPE ColourData DECLARE ColourCode : STRING DECLARE Colour : Spectrum DECLARE Wavelength : INTEGER DECLARE Frequency : REAL DECLARE PrimaryColour : BOOLEAN ENDTYPE</pre>

2(a)(i)	1 mark each - Class header (and end) - Declaration of 2 private attributes with correct data types - Constructor header (and end) within class with 2 parameters assigning parameters to attributes Example program code Java <pre>class Train{ private String Number; private Integer Route; public Train(String pNumber, Integer pRoute){ Number = pNumber; Route = pRoute; } }</pre> VB.NET <pre>Class Train Private TrainIDNumber As String Private Route As Integer Sub New(pNumber, pRoute) TrainIDNumber = pNumber Route = pRoute End Sub End Class</pre> Python <pre>class Train(): def __init__(self, pNumber, pRoute): self.__TrainIDNumber = pNumber #string self.__Route = pRoute #integer</pre>
2(a)(ii)	1 mark each - One get method header (and end) with no parameter returning correct attribute - Second correct get method Example program code Java <pre>public String GetTrainNumber(){ return Number; } public Integer GetRoute(){ return Route; }</pre> VB.NET <pre>Function GetTrainIDNumber() Return TrainIDNumber End Function Function GetRoute() Return Route End Function</pre> Python <pre>def GetTrainIDNumber(self): return self.__TrainIDNumber def GetRoute(self): return self.__Route</pre>

2(b)	1 mark each - One instance of train with correct arguments and stored in a variable/structure - Remaining three instances correct Example program code Java <pre>Train FirstTrain = new Train("12ADV", 134); Train SecondTrain = new Train("33ART", 20); Train ThirdTrain = new Train("9FKF", 3); Train FourthTrain = new Train("21VBC", 24)</pre> VB.NET <pre>Dim FirstTrain As Train = New Train("12ADV", 134) Dim SecondTrain As Train = New Train("33ART", 20) Dim ThirdTrain As Train = New Train("9FKF", 3) Dim FourthTrain As Train = New Train("21VBC", 24)</pre> Python <pre>FirstTrain = Train("12ADV",134) SecondTrain = Train("33ART",20) ThirdTrain = Train("9FKF",3) FourthTrain = Train("21VBC",24)</pre>
------	--

2(c)(i)	1 mark each - Class header (and end) with four private attributes with appropriate data types - Constructor header (and end) within class taking 2 parameters assigning parameters to attributes, initialising NumberTrains to 0, initialising Trains to an empty array Example program code Java <pre>class Station{ private String StationID; private Integer NumberPlatforms; private Train[] Trains = new Train[10]; private Integer NumberTrains; public Station(String pID, Integer pNumberOfPlatforms){ StationID = pID; NumberPlatforms = pNumberOfPlatforms; NumberTrains = 0; } }</pre> VB.NET <pre>Class Station Private StationID As String Private NumberPlatforms As Integer Private Trains(9) As Train Private NumberTrains As Integer Sub New(pID, pNumberOfPlatforms) StationID = pID NumberPlatforms = pNumberOfPlatforms NumberTrains = 0 End Sub End Class</pre>
---------	--

2(c)(i)	Python <pre>class Station(): def __init__(self, pID, pNumberOfPlatforms): self.__StationID = pID #string self.__NumberPlatforms = pNumberOfPlatforms #integer self.__Trains = [] #train 10 elements self.__NumberTrains = 0 #integer</pre>
---------	--

2(c)(ii)	1 mark each - Method header (and close) taking one Train parameter - Checking if all platforms are full and returning FALSE (Otherwise) Storing parameter in array Trains incrementing NumberTrains and returning True Example program code Java <pre>public Boolean AddTrain(Train NewTrain){ if(NumberTrains >= NumberPlatforms){ return false; } Trains[NumberTrains] = NewTrain; NumberTrains++; return true; }</pre> VB.NET <pre>Function AddTrain(NewTrain) If NumberTrains >= NumberPlatforms Then Return False End If Trains(NumberTrains) = NewTrain NumberTrains = NumberTrains + 1 Return True End Function</pre> Python <pre>def AddTrain(self, NewTrain): if self.__NumberTrains >= self.__NumberPlatforms: return False else: self.__Trains.append(NewTrain) self.__NumberTrains += 1 return True</pre>
----------	---

2(c)(iii)	1 mark each - Method header (and close) and returning a string in all cases - Checking if no trains and returning "There are no trains" (Otherwise) Looping through each train in the station accessing train ID number and route number using get methods ... creating a string with ID number and route number for each train ... returning correctly formatted string Example program code Java <pre>public String GetTrains() { if (NumberTrains == 0) { return "There are no trains"; } String OutputLine = "The trains at station " + StationID + " are: \n"; for (Integer x = 0; x < NumberTrains; x++) { OutputLine = OutputLine + Trains[x].GetTrainNumber() + " on route number " + Trains[x].GetRoute() + "\n"; } return OutputLine; }</pre> VB.NET <pre>Function GetTrains() If NumberTrains = 0 Then Return "There are no trains" End If Dim OutputLine As String = "The trains at station " & StationID & " are:" & vbNewLine For x = 0 To NumberTrains - 1 OutputLine = OutputLine & Trains(x).GetTrainIDNumber() & " on route number " & Trains(x).GetRoute() & vbNewLine Next Return OutputLine End Function</pre>
2(c)(iii)	Python <pre>def GetTrains(self): if self.__NumberTrains == 0: return "There are no trains" OutputLine = "The trains at station " + self.__StationID + " are: \n" for x in range(self.__NumberTrains): OutputLine = OutputLine + self.__Trains[x].GetTrainIDNumber() + " on route number " + str(self.__Trains[x].GetRoute()) + "\n" return OutputLine</pre>
2(d)(i)	1 mark each - One instance of Station created with correct arguments and stored - Second correct instance and stored Example program code Java <pre>Station SouthStation = new Station("STH", 2); Station NorthStation = new Station("NTH", 1);</pre> VB.NET <pre>Dim SouthStation As Station = New Station("STH", 2) Dim NorthStation As Station = New Station("NTH", 1)</pre> Python <pre>SouthStation = Station("STH", 2) NorthStation = Station("NTH", 1)</pre>

2(d)(ii)	1 mark each - Calling AddTrain for 3 correct trains for station STH once - Calling AddTrain for 1 correct train for station NTH once - Outputting "Station is full" if any return value is FALSE - Calling GetTrains() for both stations and outputting return values Example program code Java <pre>Boolean ReturnValue = SouthStation.AddTrain(FirstTrain); if (ReturnValue == false) { System.out.println("Station is full"); } ReturnValue = SouthStation.AddTrain(SecondTrain); if (ReturnValue == false) { System.out.println("Station is full"); } ReturnValue = SouthStation.AddTrain(ThirdTrain); if (ReturnValue == false) { System.out.println("Station is full"); } ReturnValue = NorthStation.AddTrain(FourthTrain); if (ReturnValue == false) { System.out.println("Station is full"); } System.out.println(SouthStation.GetTrains()); System.out.println(NorthStation.GetTrains());</pre> VB.NET <pre>Dim ReturnValue As Boolean = SouthStation.AddTrain(FirstTrain) If ReturnValue = False Then Console.WriteLine("Station is full") End If ReturnValue = SouthStation.AddTrain(SecondTrain) If ReturnValue = False Then Console.WriteLine("Station is full") End If ReturnValue = NorthStation.AddTrain(FourthTrain) If ReturnValue = False Then Console.WriteLine("Station is full") End If Console.WriteLine(SouthStation.GetTrains()) Console.WriteLine(NorthStation.GetTrains())</pre> Python <pre>ReturnValue = SouthStation.AddTrain(FirstTrain) if ReturnValue == False: print("Station is full") ReturnValue = SouthStation.AddTrain(SecondTrain) if ReturnValue == False: print("Station is full") ReturnValue = SouthStation.AddTrain(ThirdTrain) if ReturnValue == False: print("Station is full") ReturnValue = NorthStation.AddTrain(FourthTrain) if ReturnValue == False: print("Station is full") print(SouthStation.GetTrains()) print(NorthStation.GetTrains())</pre>
----------	---

2(d)(iii)	1 mark each, screenshot(s) showing: • One output of "Station is full" • Output of correct data for both stations (in correct format) e.g. <pre>Station is full The trains at station STH are: 12ADV on route number 134 33ART on route number 20 The trains at station NTH are: 21VBC on route number 24</pre>
-----------	---

1(a)	One mark per mark point MP1 TYPE Vehicle = MP2 (M100, M230, T101, T102, T120, T150) Example answer: <pre>TYPE Vehicle = (M100, M230, T101, T102, T120, T150)</pre>
1(b)	One mark per mark point MP1 TYPE Booking and ENDTYPE correct MP2 Declare used correctly for every field in the response MP3 Any four fields correct MP4 Remaining fields correct Example answer: <pre>TYPE Booking DECLARE BookingNumber : STRING DECLARE Destination : STRING DECLARE ClientName : STRING DECLARE ClientTelephone : STRING DECLARE DateOfDeparture : DATE DECLARE PickupAddress : STRING DECLARE TaxiUsed : Vehicle ENDTYPE</pre>

1(a)	One mark for each mark point MP1 Use of correct Flight1 variable with given field names MP2 Correct assignments of four string data values MP3 Correct assignment of date data value Example answer <pre>Flight1.FlightNumber ← "SB2789" Flight1.Destination ← "Dublin" Flight1.FlightDate ← 30/07/2025 Flight1.Gate ← "N03" Flight1.Airline ← "Cambridge Airways"</pre>
1(b)(i)	One mark per mark point MP1 TYPE GateID = MP2 (N01, N02, N03, W01, W02, W03, W04) Example answer <pre>TYPE GateID = (N01, N02, N03, W01, W02, W03, W04)</pre>
1(b)(ii)	<pre>DECLARE Gate : GateID</pre>

1(a)	One mark per mark point MP1 TYPE VideoLibrary and ENDTYPE correct MP2 Declare used correctly for every field in the response MP3 All Three STRING fields correct (shaded) MP4 Remaining Three fields correct (unshaded) Example answer <pre>TYPE VideoLibrary DECLARE VideoID : STRING DECLARE Title : STRING DECLARE ReleaseYear : INTEGER DECLARE PurchaseDate : DATE DECLARE VideoFormat : STRING DECLARE RunningTime : INTEGER ENDTYPE</pre>
1(b)	One mark for identification of field and one mark for reason VideoFormat This field can have a fixed range of possible values

9(a)	To ensure that the attributes are only accessible using the class's own methods/within the class.
9(b)	One mark per mark point (Max 5) MP1 Two correct attributes with sensible names and correct data types. MP2 Constructor present. MP3 Two correct setters with exact names and appropriate parameters and data types. MP4 Two correct getters with appropriate names. MP5 Name assigned to pet name getter matches the attribute. <pre> class Pet { PetID : STRING PetType : STRING OwnerTelephone : STRING DateRegistered : DATE PetName : STRING OwnerName : STRING Constructor () SetPetID (APetID : STRING) SetDateRegistered (RegDate : DATE) GetPetName () GetOwnerTelephone () }</pre>

3(a)	One mark per mark point (Max 3) MP1 A user-defined record data type is a composite data type MP2 It uses other data types in its definition to form a single new data type MP3 The data types referenced may be primitive data types from a programming language or they may be other user-defined data types. MP4 Includes related items MP5 Includes a fixed number of items.
3(b)	One mark for TYPE Order and ENDTYPE correct One mark for correct use of DECLARE in all declarations seen One mark for the two shaded declarations One mark for the two unshaded declarations Example answer TYPE Order <pre> DECLARE AccountNumber : INTEGER DECLARE OrderNumber : INTEGER DECLARE OrderPrice : REAL DECLARE OrderDate : DATE ENDTYPE </pre>

6(a)	One mark per mark point (Max 3) MP1 A set user-defined data type is a composite data type MP2 ... which includes a list of unordered elements MP3 Set theory operations, such as intersection and union, can be applied to these elements MP4 A set data type includes the type of data/data type it uses as part of its definition MP5 All the elements are of the same data type.
6(b)	One mark for each mark point (Max 4) MP1 TYPE SymbolSet/Operators = MP2 SET OF CHAR MP3 DEFINE Operators/SymbolSet MP4 ('+', '-', '*', '/', '^') MP5 : SymbolSet/Operators Example answers <pre> TYPE SymbolSet = SET OF CHAR DEFINE Operators ('+', '-', '*', '/', '^') : SymbolSet TYPE Operators = SET OF CHAR DEFINE SymbolSet ('+', '-', '*', '/', '^') : Operators </pre>