Question 1:

In this question, you will draw a class hierarchy chart that best defines the relationships of the following **Vehicles** classes:

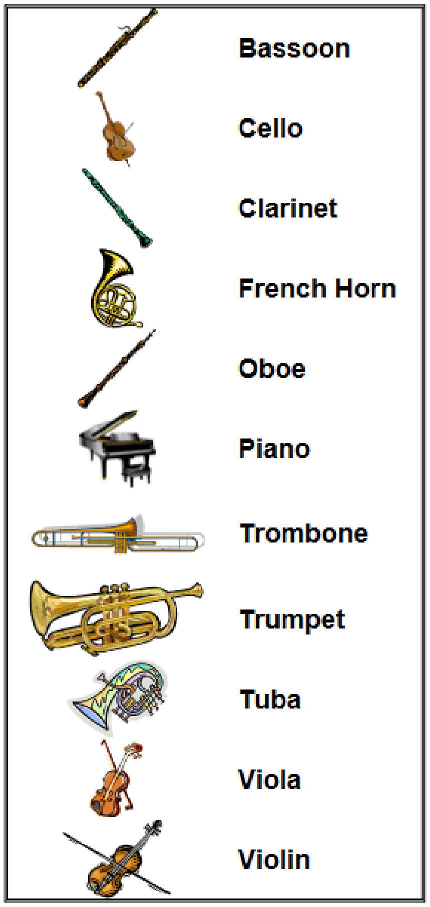
Think of the different vehicle groups there are. Although there may be several ways this chart could be drawn, full marks will be given for the best-designed solution. There should be **at least** three layers.

You may need to create one or more additional abstract classes, but you do not need to create additional abstract classes for only one child class.

You may use whichever tool you wish to create your hierarchy chart: Word, Visio, Umlet, even hand-drawn and photographed by your camera as long as it's readable. ~~Upload your file to this question.~~

Question 1:

In this question, you will draw a class hierarchy chart that best defines the relationships of the following **Musical Instruments** classes:



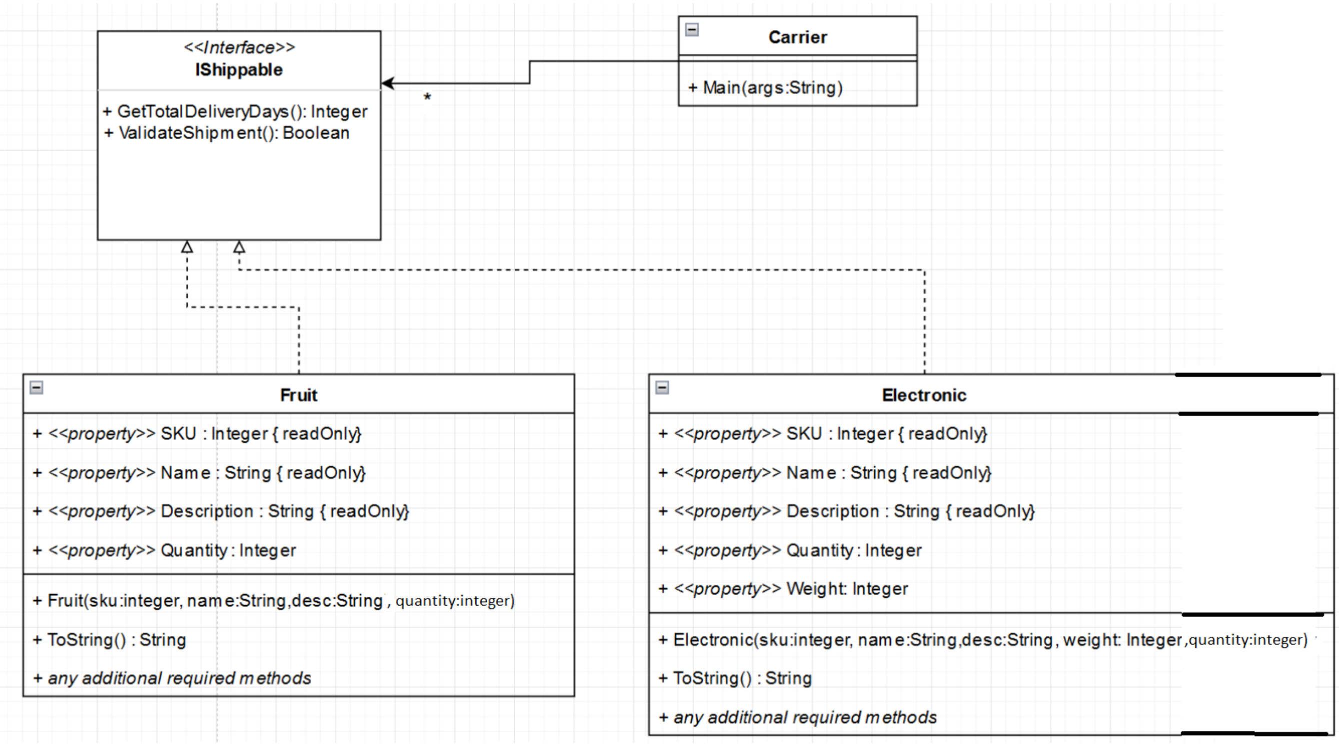
Think of the different musical groups there are. Although there may be several ways this chart could be drawn, full marks will be given for the best-designed solution. There should be **at least** three layers.

You may need to create one or more additional abstract classes, but you do not need to create additional abstract classes for only one child class.

You may use whichever tool you wish to create your hierarchy chart: Word, Visio, Umlet, even hand-drawn and photographed by your camera as long as it's readable. ~~Upload your file to this question.~~

Question 2

You run Carrier Shipping company that ships fruits and electronics. The receptionist at your company is often required to quickly determine the delivery days. Look over the following UML diagram carefully (marks are shown for each entity):

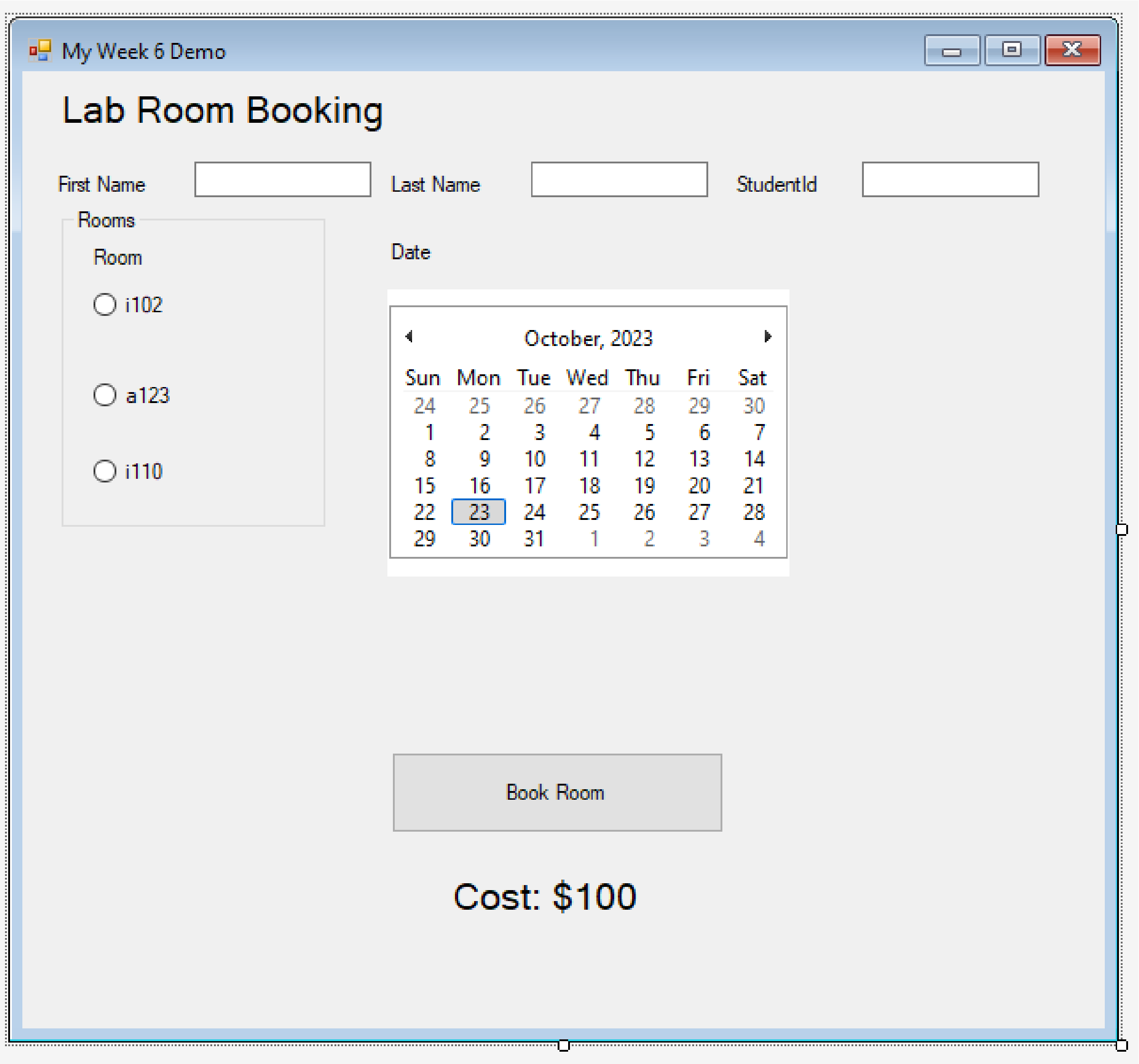


<<Interface>>
IShippable
+ GetTotalDeliveryDays(): Integer
+ ValidateShipment(): Boolean
*
Carrier
+ Main(args:String)
Fruit
+ <<property>> SKU : Integer { readOnly}
+ <<property>> Name : String { readOnly}
+ <<property>> Description : String { readOnly}
+ <<property>> Quantity : Integer
+ Fruit(sku:integer, name:String,desc:String , quantity:integer)
+ ToString() : String
+ any additional required methods
Electronic
+ <<property>> SKU : Integer { readOnly}
+ <<property>> Name : String { readOnly}
+ <<property>> Description : String { readOnly}
+ <<property>> Quantity : Integer
+ <<property>> Weight: Integer
+ Electronic(sku:integer, name:String,desc:String, weight: Integer,quantity:integer)
+ ToString() : String
+ any additional required methods

You are to write the code for all the shown components, obeying the following guidelines:

- Do not write unnecessary code. All methods and instance variables are in the UML.
- Your classes must make use of reusable code where possible. Use best practices.
- The **Fruit** and **Electronic** constructor must throw appropriate exception (with custom message) if the sku is not provided.
- The **Fruit** constructor must throw appropriate exception if quantity is greater than 100.
- The **Electronic** constructor must throw appropriate exception if weight is 0 or less than 0.
- Shipment details
  - **Fruits** takes ½ day per quantity.
  - **Electronics** takes 1 day per quantity if less weight less than 10; everything over 10 is 2 days per quantity.
  - A shipment is only valid if there is a SKU and Quantity
    - And for **Electronics** there must also be weight
- The **Carrier** class must use a single List<> to store all polymorphic objects.
- The **Carrier** class must output <u>each</u> Item (ie. “10 Apples will take 5 days”)
- The **Carrier** class must output the total Shipment Days for everything in collection (“Everything will take 201 days”)
- Hardcode the following object data into your **Main**() method (no user input is required)
  - Fruit: [sku:20, name:”Apple” , desc:”Red Apple”, quantity:10]
  - Electronic: [sku:30, name:”Iphone”, desc:”Iphone x”, quantity:30, weight:”10”]
  - Electronic: [sku:40, name:”TV”,desc:”85” TV”, quantity:5, weight:”200”]
  - Fruit: [sku:50, name:”Lemon”,desc:”Yellow Lemon”, quantity:6]
  - Fruit: [sku:51, name:”Grape”,desc:”Green Grape”, quantity:8]
  - Electronic: [sku:80, name:”Laptop”,desc:”Thinkpad”, quantity:9, weight:”50”]
  - Electronic: [sku:82, name:”Keyboard”,desc:”keys”, quantity:2, weight:”5”]
  - Fruit: [sku:96, name:”Pear”,desc:”Brown Pear”, quantity:80]

Question 3



October, 2023

Sun	Mon	Tue	Wed	Thu	Fri	Sat
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Cost: $100

- The Window should be properly titled.
- The “Lab Room booking” label at top should be 16pt; The “Cost: $X” label should be 16pt; all other labels is default size
- There must be one control group for rooms
- There can only one room selected at a time
- You can assume that the user will only select one day in month calendar(never multiple days)
- Cost:
  - i102 & i110 Monday and Tuesday, $10 per day; all other days $5 per day
  - a123, all days, $7 day

- Clicking Book Room calculates cost and displays it in a Cost label