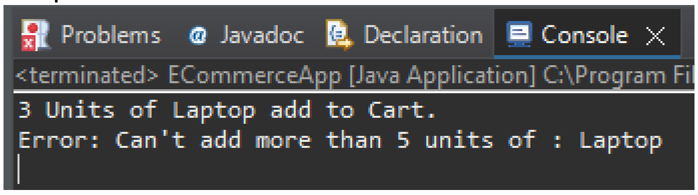
Java Programming

Practical Assignments 3

1) You have developed an e-commerce website for your client. The maximum no of units of a single prodcut that one user can add to the cart is 5. If the user adds more than 5 units of a single product, then your application is expected to throw, **MaximumProductsLimitExceededException**. Write a custom exception class to achieve this.

Code=>

```
//
package exceptionProduct;

public class MaximumProductsLimitExceededException extends Exception{

	public MaximumProductsLimitExceededException(String message) {
		super(message);
	}

}
//
package exceptionProduct;

public class Product {
	private String name;
	private double price;

	public Product(String name,double price) {
		this.name=name;
		this.price=price;
	}

	public String getname() {
		return name;
	}
	public double getprice() {
		return price;
	}

}
//
package exceptionProduct;

public class Cart {
	private Product product;
	private int quantity;

	public Cart(Product product) {
		this.product=product;
		this.quantity=0;
	}

	public void addProduct(int quantity)throws
MaximumProductsLimitExceededException{
		if(this.quantity + quantity>5) {
			throw new MaximumProductsLimitExceededException("Can't add
more than 5 units of : "+product.getname());
		}else {
			this.quantity += quantity;
			System.out.println(quantity+" Units of "+product.getname()+" add to
Cart.");
		}
	}

	public int getquantity() {
		return quantity;
	}
}
//
package exceptionProduct;

public class ECommerceApp {
	public static void main(String[] args) {
		Product p=new Product("Laptop",500);
		Cart c=new Cart(p);

		try {
			c.addProduct(3);
			c.addProduct(4);
		}
		catch(MaximumProductsLimitExceededException e){
			System.out.println("Error: "+e.getMessage());
		}

	}
}
```
Output=>



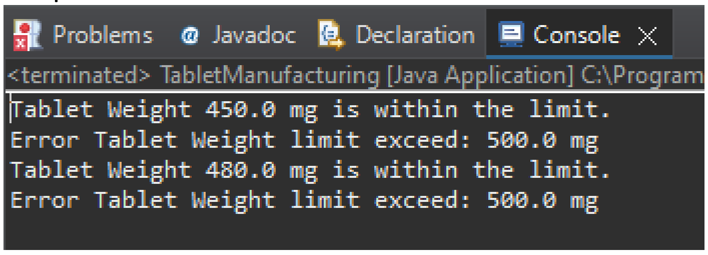
```
<terminated> ECommerceApp [Java Application] C:\Program Fi
3 Units of Laptop add to Cart.
Error: Can't add more than 5 units of : Laptop
```

2) The manufacturing of your medical company has very strict standards of product specifications. After each pill /tablet is ready, it is weighed. If the weight of the tablet exceeds the allowed limit, **TabletWeightExceededException** is raised. Using exception handling in Java, write the program to achieve the above business requirement.

Code=>

```
//
package weight;

public class TabletWeightExceededException extends Exception{
	public  TabletWeightExceededException(String message) {
		super(message);
	}
}
//
package weight;

public class TabletManufacturing {
	private static final double ALLOWED_WEIGHT_LIMIT = 500.0;

	public static void checkTabletWeight(double tabletWeight)throws
TabletWeightExceededException{
		if(tabletWeight>ALLOWED_WEIGHT_LIMIT) {
			throw new TabletWeightExceededException("Tablet Weight limit
exceed: "+ALLOWED_WEIGHT_LIMIT+" mg");
		}
	}

	public static void processTablets(double[] tabletWeights) {
		for(double weight: tabletWeights) {
			try {
				checkTabletWeight(weight);
				System.out.println("Tablet Weight "+weight+" mg is within the
limit.");
			}catch(TabletWeightExceededException e) {
				System.out.println("Error "+e.getMessage());
			}
		}
	}

	public static void main(String[] args) {
		double[] tabletWeights= {450.0,510.0, 480.0, 550.0};
		processTablets(tabletWeights);
	}
}
```

Output=>



```
<terminated> TabletManufacturing [Java Application] C:\Program
Tablet Weight 450.0 mg is within the limit.
Error Tablet Weight limit exceed: 500.0 mg
Tablet Weight 480.0 mg is within the limit.
Error Tablet Weight limit exceed: 500.0 mg
```

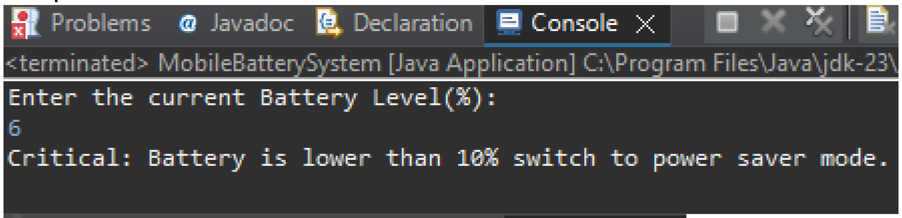
3) When the battery of your mobile phone is less than 20%, the system should generate, **LowBatteryException** to alert the user to start charging the device. If the battery goes lower than 10 then the system should raise InsufficientChargeException and put the unit on power saver mode. Using exception handling in Java, write the program to achieve the above business requirement.

Code=>

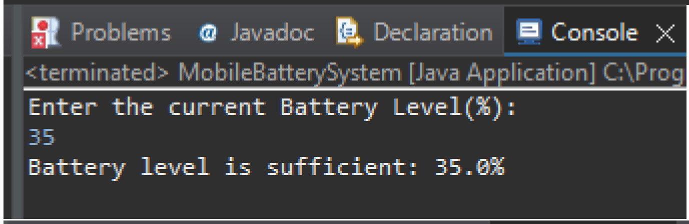
```
//
package battery;

public class InsufficientChargeException extends Exception{
	public InsufficientChargeException(String message) {
		super(message);
	}
}
//
package battery;

public class LowBatteryException extends Exception {
	public LowBatteryException(String message) {
		super(message);
	}
}
//
package battery;
import java.util.Scanner;

public class MobileBatterySystem {
	public static void checkBatteryLevel(double BatteryLevel)throws
LowBatteryException, InsufficientChargeException{
		if(BatteryLevel<10) {
			throw new InsufficientChargeException("Battery is lower than 10%
switch to power saver mode.");
		}else if(BatteryLevel<20){
			throw new LowBatteryException("Battery is below 20%. Please start
charging your device.");
		}
	}
	public static void main(String[] args) {
		Scanner sc=new Scanner(System.in);
```

```
        System.out.println("Enter the current Battery Level(%): ");
        double BatteryLevel =sc.nextDouble();

        try {
            checkBatteryLevel(BatteryLevel);
            System.out.println("Battery level is sufficient: " + BatteryLevel + "%");
        }catch(LowBatteryException e){
            System.out.println("Warning "+e.getMessage());
        }catch(InsufficientChargeException e) {
            System.out.println("Critical: "+e.getMessage());
        }
        sc.close();
    }
}
```

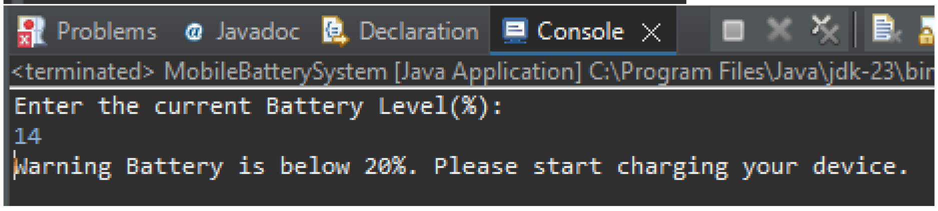
Output=>



```
<terminated> MobileBatterySystem [Java Application] C:\Program Files\Java\jdk-23\
Enter the current Battery Level(%):
6
Critical: Battery is lower than 10% switch to power saver mode.
```



```
<terminated> MobileBatterySystem [Java Application] C:\Prog
Enter the current Battery Level(%):
35
Battery level is sufficient: 35.0%
```



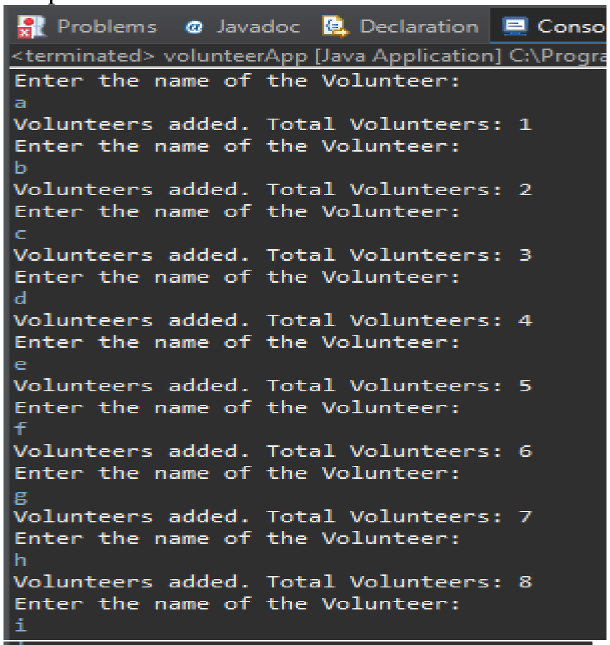
```
<terminated> MobileBatterySystem [Java Application] C:\Program Files\Java\jdk-23\bir
Enter the current Battery Level(%):
14
Warning Battery is below 20%. Please start charging your device.
```

4) You are writing an app for taking names of the volunteers for Cultural Committee of your Institute. According to the guidelines only 15 members are allowed in the committee. Using your app, take the names of the interested candidates till the number reaches 15. Once the threshold is crossed, display a message, “No more candidates allowed as volunteers. Thank you”. Use ArrayList to achieve the above given business logic

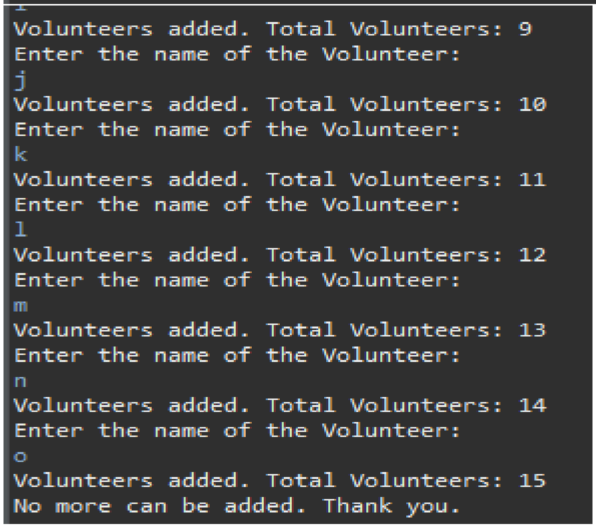
Code=>

```
//
package App;
import java.util.ArrayList;
import java.util.Scanner;

public class volunteerApp {
    public static void main(String[] args) {
        ArrayList<String> volunteers = new ArrayList<>();

        Scanner sc = new Scanner(System.in);

        final int MAX_VOLUNTEERS=15;
```

```
                while(volunteers.size()<MAX_VOLUNTEERS) {
                        System.out.println("Enter the name of the Volunteer: ");
                        String name=sc.nextLine();

                        volunteers.add(name);

                        System.out.println("Volunteers added. Total Volunteers: "+volunteers.size());

                        if(volunteers.size()==MAX_VOLUNTEERS) {
                                System.out.println("No more can be added. Thank you.");
                                break;
                        }
                }
                sc.close();
        }
}
```
Output=>



```
<terminated> volunteerApp [Java Application] C:\Progra
Enter the name of the Volunteer:
a
Volunteers added. Total Volunteers: 1
Enter the name of the Volunteer:
b
Volunteers added. Total Volunteers: 2
Enter the name of the Volunteer:
c
Volunteers added. Total Volunteers: 3
Enter the name of the Volunteer:
d
Volunteers added. Total Volunteers: 4
Enter the name of the Volunteer:
e
Volunteers added. Total Volunteers: 5
Enter the name of the Volunteer:
f
Volunteers added. Total Volunteers: 6
Enter the name of the Volunteer:
g
Volunteers added. Total Volunteers: 7
Enter the name of the Volunteer:
h
Volunteers added. Total Volunteers: 8
Enter the name of the Volunteer:
i
```



```
Volunteers added. Total Volunteers: 9
Enter the name of the Volunteer:
j
Volunteers added. Total Volunteers: 10
Enter the name of the Volunteer:
k
Volunteers added. Total Volunteers: 11
Enter the name of the Volunteer:
l
Volunteers added. Total Volunteers: 12
Enter the name of the Volunteer:
m
Volunteers added. Total Volunteers: 13
Enter the name of the Volunteer:
n
Volunteers added. Total Volunteers: 14
Enter the name of the Volunteer:
o
Volunteers added. Total Volunteers: 15
No more can be added. Thank you.
```

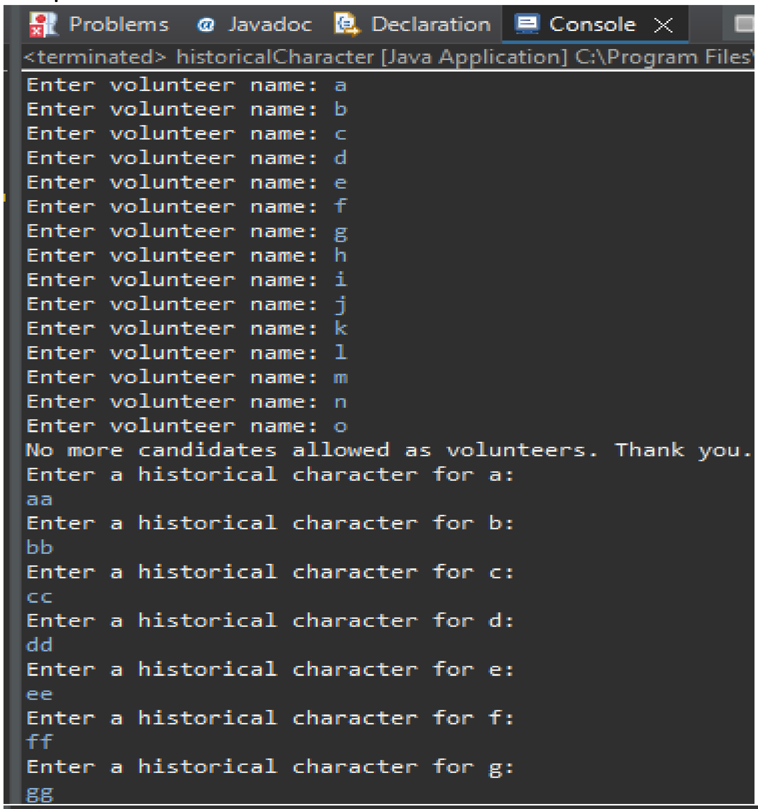
.

5) Once the above list of volunteers is finalized, each volunteer needs to pick a historical character as his/her badge icon. Using your app, take the name of the historical character from the volunteers and store them for future uses. Also, no two characters should be the same. In case the character is already in the list, ask the volunteer to enter some other character. Use ArrayList to achieve the above given requirement.

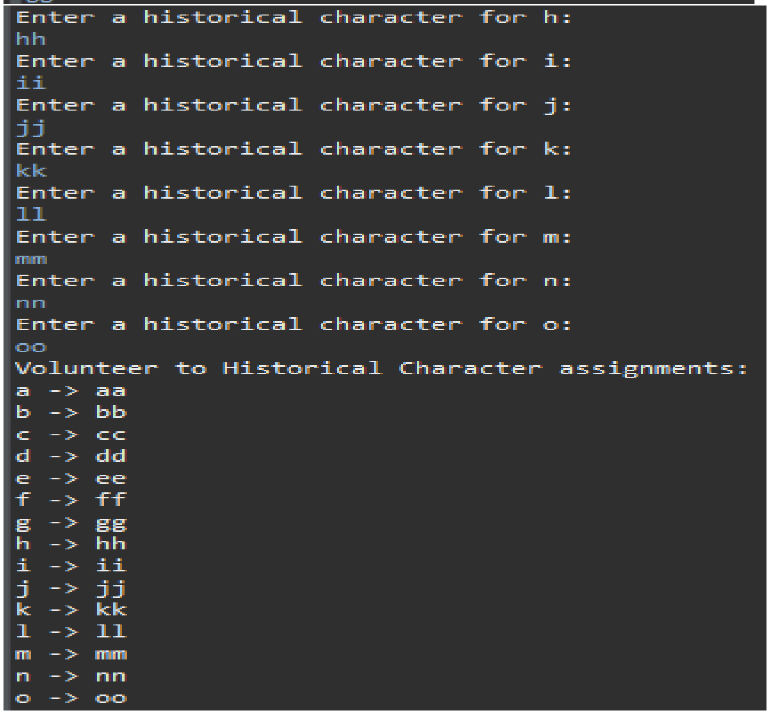
Code=>

```
//
package App;
import java.util.ArrayList;
import java.util.Scanner;

public class historicalCharacter {

        public static void main(String[] args) {
          Scanner sc = new Scanner(System.in);

          ArrayList<String> volunteers = new ArrayList<>();
          ArrayList<String> historicalCharacters = new ArrayList<>();
          int maxMembers = 15;

          // Collect names of volunteers
          while (volunteers.size() < maxMembers) {
            System.out.print("Enter volunteer name: ");
            String name = sc.nextLine();
            volunteers.add(name);
          }

          // Once we have 15 members, show a message
          System.out.println("No more candidates allowed as volunteers. Thank you.");

          for(String volunteer: volunteers) {
              boolean valid =false;

              while(!valid) {
                      System.out.println("Enter a historical character for " + volunteer + ":
");
                      String character= sc.nextLine();

                      if(historicalCharacters.contains(character)) {
                              System.out.println("This character is already taken. Please
choose another one.");
                      }else {
                              historicalCharacters.add(character);
                              valid =true;
                      }
              }
          }
```

```
                System.out.println("Volunteer to Historical Character assignments:");
                for(int i=0;i<volunteers.size();i++) {
                    System.out.println(volunteers.get(i) + " -> " + historicalCharacters.get(i));
                }
           }

}
```
Output=>



```
<terminated> historicalCharacter [Java Application] C:\Program Files\
Enter volunteer name: a
Enter volunteer name: b
Enter volunteer name: c
Enter volunteer name: d
Enter volunteer name: e
Enter volunteer name: f
Enter volunteer name: g
Enter volunteer name: h
Enter volunteer name: i
Enter volunteer name: j
Enter volunteer name: k
Enter volunteer name: l
Enter volunteer name: m
Enter volunteer name: n
Enter volunteer name: o
No more candidates allowed as volunteers. Thank you.
Enter a historical character for a:
aa
Enter a historical character for b:
bb
Enter a historical character for c:
cc
Enter a historical character for d:
dd
Enter a historical character for e:
ee
Enter a historical character for f:
ff
Enter a historical character for g:
gg
```



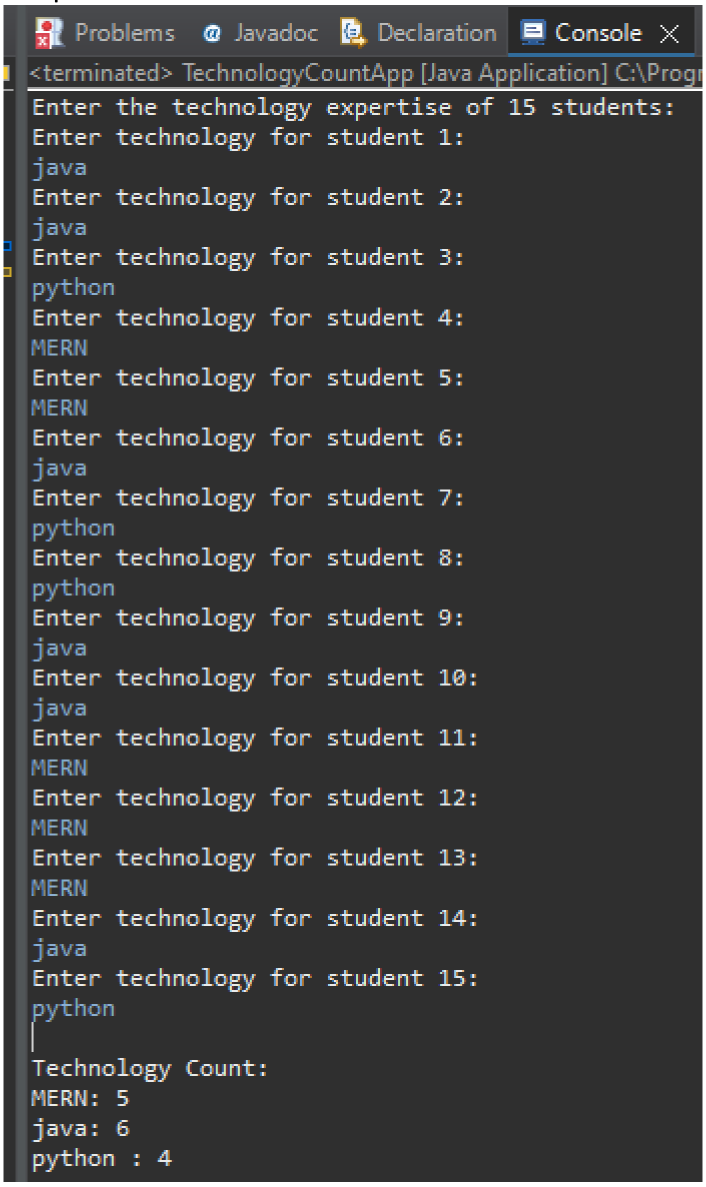
```
Enter a historical character for h:
hh
Enter a historical character for i:
ii
Enter a historical character for j:
jj
Enter a historical character for k:
kk
Enter a historical character for l:
ll
Enter a historical character for m:
mm
Enter a historical character for n:
nn
Enter a historical character for o:
oo
Volunteer to Historical Character assignments:
a -> aa
b -> bb
c -> cc
d -> dd
e -> ee
f -> ff
g -> gg
h -> hh
i -> ii
j -> jj
k -> kk
l -> ll
m -> mm
n -> nn
o -> oo
```

6) The placement cell of your Institute has asked you to share the name of one technology which you are expert in. Using an app, take this from 15 students. The cell then wants you to give a technology count based on the input. For ex, how many students chose Java, how many chose Python, how many entered MERN, etc. Demonstrate the use of ArrayList to achieve this.

Code=>

```
//
package tandPCell;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class TechnologyCountApp {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner sc = new Scanner(System.in);

		ArrayList<String> technologies =new ArrayList<>();

		int totalStudents=15;

		System.out.println("Enter the technology expertise of 15 students:");
		for(int i=0;i<totalStudents;i++) {
			System.out.println("Enter technology for student " + (i + 1) + ": ");
			String tech =sc.nextLine();
			technologies.add(tech);
		}

		Collections.sort(technologies);

		String currentTechnology=null;
		int count=0;

		 System.out.println("\nTechnology Count:");

		 for(String tech:technologies) {
			 if(tech.equals(currentTechnology)) {
				 count++;
			 }else {
				 if(currentTechnology != null) {
					 System.out.println(currentTechnology + ": " + count);
				 }
				 currentTechnology = tech;
				 count=1;
			 }
		 }

		 if(currentTechnology != null) {
```

```
                                        System.out.println(currentTechnology +" : "+count);
                                }
                }

}
```
Output=>



```
Problems  Javadoc  Declaration  Console
<terminated> TechnologyCountApp [Java Application] C:\Progr
Enter the technology expertise of 15 students:
Enter technology for student 1:
java
Enter technology for student 2:
java
Enter technology for student 3:
python
Enter technology for student 4:
MERN
Enter technology for student 5:
MERN
Enter technology for student 6:
java
Enter technology for student 7:
python
Enter technology for student 8:
python
Enter technology for student 9:
java
Enter technology for student 10:
java
Enter technology for student 11:
MERN
Enter technology for student 12:
MERN
Enter technology for student 13:
MERN
Enter technology for student 14:
java
Enter technology for student 15:
python

Technology Count:
MERN: 5
java: 6
python : 4
```

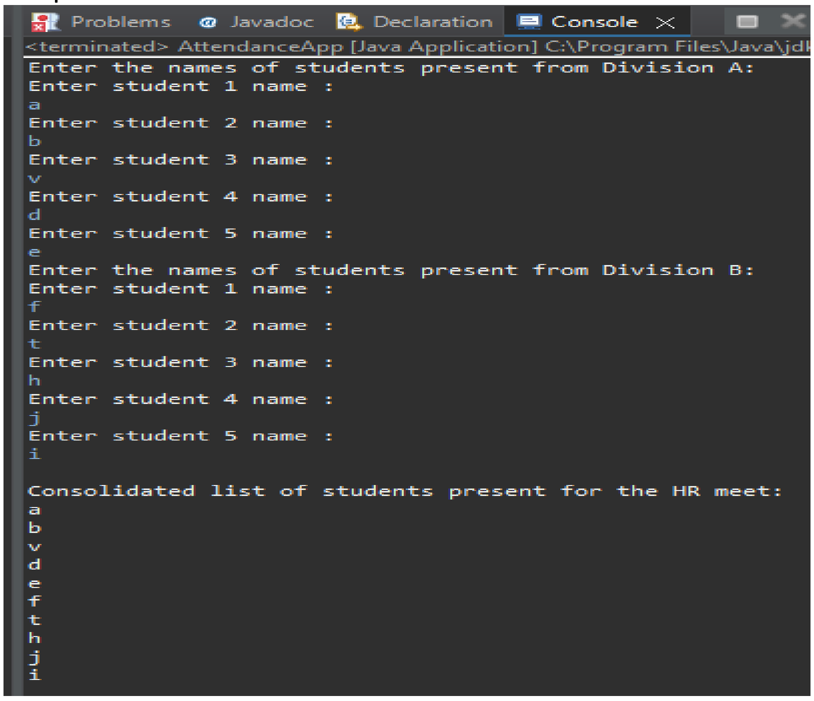
7) For the recently held HR meet, the CR and LR of division A and B marked the attendance for their respective classes in separate lists. The TPO cell wants a consolidated list of FYMCA students who were present for the event. Write a program using ArrayList to mark division wise attendance first and then give the consolidated list.

Code=>

```
//
package attendance;
import java.util.ArrayList;
import java.util.Scanner;
public class AttendanceApp {
        public static void main(String[] args) {
```

```
                    Scanner sc= new Scanner(System.in);
                    ArrayList<String> divA=new ArrayList<>();
                    ArrayList<String> divB=new ArrayList<>();
                    int divAsize=5;
                    int divBsize=5;
                     System.out.println("Enter the names of students present from Division A:");
                     for(int i=0;i<divAsize;i++) {
                               System.out.println("Enter student "+( i + 1)+" name : ");
                               String name =sc.nextLine();
                               divA.add(name);
                     }
                     System.out.println("Enter the names of students present from Division B:");
                     for(int i=0;i<divBsize;i++) {
                               System.out.println("Enter student "+( i + 1)+" name : ");
                               String name =sc.nextLine();
                               divB.add(name);
                     }
                     ArrayList<String> consolidatedList = new ArrayList<>(divA);
                     consolidatedList.addAll(divB);
                     System.out.println("\nConsolidated list of students present for the HR
meet:");
                     for(String student : consolidatedList) {
                               System.out.println(student);
                     }
                                        sc.close();
          }
}
```

Output=>



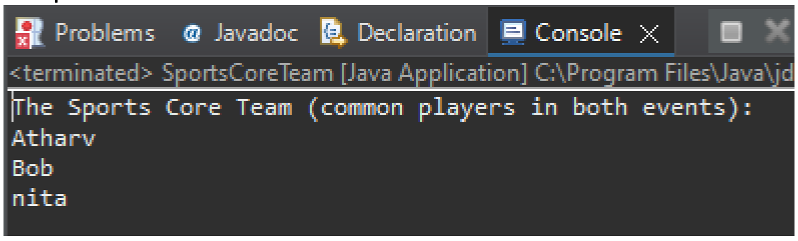
```
<terminated> AttendanceApp [Java Application] C:\Program Files\Java\jdk
Enter the names of students present from Division A:
Enter student 1 name :
a
Enter student 2 name :
b
Enter student 3 name :
v
Enter student 4 name :
d
Enter student 5 name :
e
Enter the names of students present from Division B:
Enter student 1 name :
f
Enter student 2 name :
t
Enter student 3 name :
h
Enter student 4 name :
j
Enter student 5 name :
i

Consolidated list of students present for the HR meet:
a
b
v
d
e
f
t
h
j
i
```

8) Sports cell of the Institute needs to choose its core team from those students who participated in the recently held sports events. For this, the sports coordinator has decided to consider the participants of Football and Cricket. Only those players who participated in BOTH these games will be considered for the core team. Using ArrayList, write a Java program which will take the names of the students participating in Football and Cricket. Find the common names in these two events and put them into a third list, called, SportsCoreTeam

Code=>

```
//
package Sports;
import java.util.ArrayList;

public class SportsCoreTeam {
        public static void main(String[] args) {

                ArrayList<String> football = new ArrayList<>();
                ArrayList<String>cricket = new ArrayList<>();

                 football.add("Atharv");
           football.add("Bob");
           football.add("nita");
           football.add("David");

           cricket.add("Bob");
           cricket.add("Atharv");
           cricket.add("Eva");
           cricket.add("nita");

           ArrayList<String> sportsCoreTeam = new ArrayList<String>();

           for(String footballs : football) {
                 if(cricket.contains(footballs)) {
                         sportsCoreTeam.add(footballs);
                 }
           }

           System.out.println("The Sports Core Team (common players in both events):");
           for(String player: sportsCoreTeam) {
                 System.out.println(player);
           }
        }
}
```

Output=>



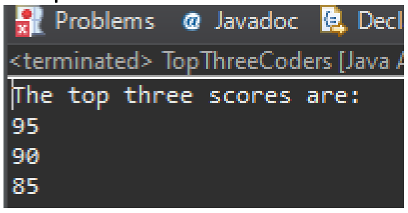
```
<terminated> SportsCoreTeam [Java Application] C:\Program Files\Java\jd
The Sports Core Team (common players in both events):
Atharv
Bob
nita
```

9) The top three scorers in the coding competition will be given a certificate and trophy by the Coding Club. Using Vector, take the final scores (out of 100) of the participating coders and find the top three using only the max() function available in Collections.

Code=>

```
//
package topScore;
import java.util.Collections;
import java.util.Vector;

public class TopThreeCoders {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		Vector<Integer> scores =new Vector<>();

		scores.add(75);
     scores.add(90);
     scores.add(85);
     scores.add(95);
     scores.add(80);
     scores.add(70);

     Vector<Integer> topthreescores =new Vector<>();

     for(int i=0;i<3;i++) {
         int maxscore= Collections.max(scores);

         topthreescores.add(maxscore);

         scores.removeElement(maxscore);
     }

     System.out.println("The top three scores are: ");

     for(int score: topthreescores) {
         System.out.println(score);
     }
	}

}
```

Output=>



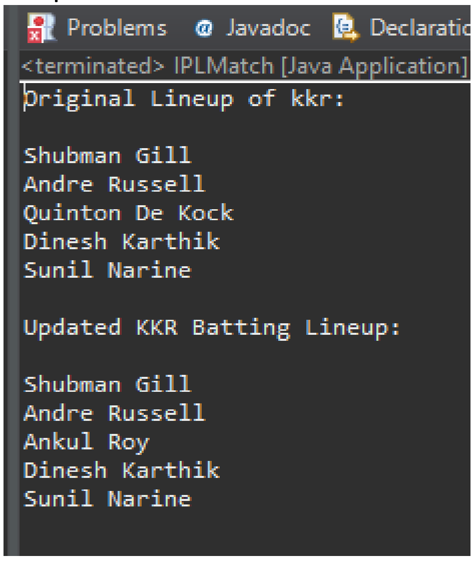
```
The top three scores are:
95
90
85
```

10) KKR and MumbaiIndians are going to play the kickstart match of this year’s IPL season. Using Vector, you have taken the names of the players in each team and are displaying the same. But there is a last minute change in the batting line up of KKR. In place of QuintonDeKock, the team will send Anukul Roy at two down position. Make this change in their batting line up and display the new order.

Code=>

```
//
package Ipl;
import java.util.Vector;

public class IPLMatch {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                        Vector<String> kkrlineup =new Vector<>();

                         kkrlineup.add("Shubman Gill");
                   kkrlineup.add("Andre Russell");
                   kkrlineup.add("Quinton De Kock"); // The player to be replaced
                   kkrlineup.add("Dinesh Karthik");
                   kkrlineup.add("Sunil Narine");


                   System.out.println("Original Lineup of kkr: \n");
                   for(String player :kkrlineup) {
                         System.out.println(player);
                   }

                   int index = kkrlineup.indexOf("Quinton De Kock");
                   if(index != -1) {
                         kkrlineup.set(index , "Ankul Roy");
                   }

                   System.out.println("\nUpdated KKR Batting Lineup: \n");
                     for (String player : kkrlineup) {
                        System.out.println(player);
                     }
        }

}
```

Output=>



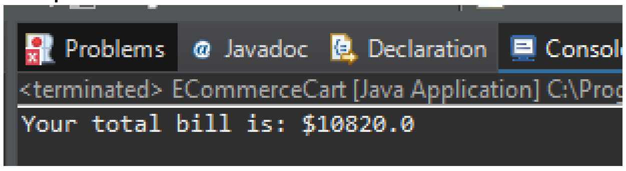
```
<terminated> IPLMatch [Java Application]
Original Lineup of kkr:

Shubman Gill
Andre Russell
Quinton De Kock
Dinesh Karthik
Sunil Narine

Updated KKR Batting Lineup:

Shubman Gill
Andre Russell
Ankul Roy
Dinesh Karthik
Sunil Narine
```

11) In the e-commerce portal designed by you, the customer adds products to the shopping cart. Use a vector to hold the objects of Product class. At the time of billing, access each product object and read its price. Add the cost of all the products and display the bill total. If the cart is empty, show a message, “Can we help you in finding what you were looking for?” and end the billing process.

Code=>

```
//
package ecommerce;


public class Product {
	String name;
	double price;

	public Product(String name,double price) {
		this.name=name;
		this.price=price;
	}

	public String getname() {
		return name;
	}

	public double getprice() {
		return price;
	}

}
//
package ecommerce;
import java.util.Vector;
```

```
public class ECommerceCart {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Vector<Product> shoppingcart = new Vector<Product>();

		shoppingcart.add(new Product("Lapto", 8320.0));
        shoppingcart.add(new Product("Smartphone", 1500.0));
        shoppingcart.add(new Product("Headphones", 1000.0));

        if(shoppingcart.isEmpty()) {
            System.out.println("Can we help you in finding what you were looking for?");
        }else {
        	double total =0.0;
        	for(Product p :shoppingcart) {
        		total +=p.getprice();
        	}
            System.out.println("Your total bill is: $" + total);
        }
	}

}
```
Output=>

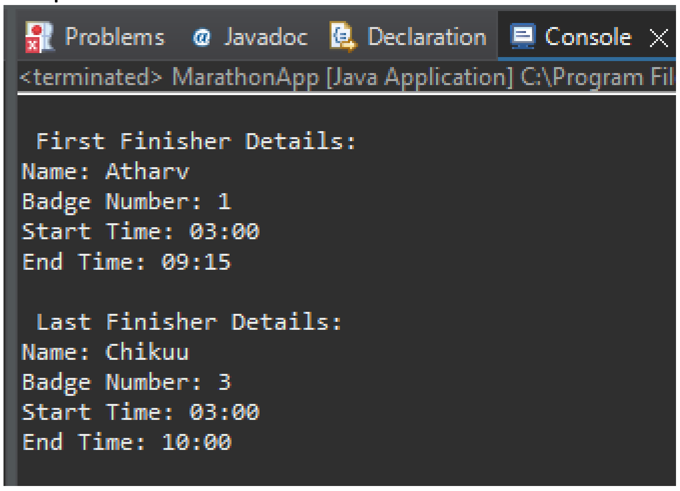


```
<terminated> ECommerceCart [Java Application] C:\Prog
Your total bill is: $10820.0
```

12) During the Marathon event the organisers maintained a list to hold the details of the finishers. Once the marathon got over, they displayed the details of the first runner to finish the marathon and the last one to finish the same. Write an app having the objects of MarathonRunner class in to a vector list, finishers. Display the details of the runner who comes first and of the who comes last. MarathonRunner class has the properties, name, badgeNbr, startTime and endTime

Code=>
```
//
package marathon;

class MarathonRunner {
    private String name;
    private int badgeNbr;
    private String startTime;
    private String endTime;

    public MarathonRunner(String name, int badgeNbr, String startTime, String endTime) {
        this.name = name;
        this.badgeNbr = badgeNbr;
        this.startTime = startTime;
        this.endTime = endTime;
    }

    public void displayDetails() {
        System.out.println("Name: " + name);
        System.out.println("Badge Number: " + badgeNbr);
        System.out.println("Start Time: " + startTime);
        System.out.println("End Time: " + endTime);
    }
}
//
package marathon;
import java.util.Vector;
public class MarathonApp {
    public static void main(String[] args) {
        Vector<MarathonRunner> finishers = new Vector<>();

        finishers.add(new MarathonRunner("Atharv", 1, "03:00", "09:15"));
        finishers.add(new MarathonRunner("Raj", 2, "03:00", "09:30"));
        finishers.add(new MarathonRunner("Chikuu", 3, "03:00", "10:00"));

        if (!finishers.isEmpty()) {
            System.out.println("\n First Finisher Details:");
            finishers.firstElement().displayDetails();

            System.out.println("\n Last Finisher Details:");
            finishers.lastElement().displayDetails();
        } else {
            System.out.println("No runners have finished yet.");
        }
    }
}
```

Output=>



```
<terminated> MarathonApp [Java Application] C:\Program Fil

 First Finisher Details:
Name: Atharv
Badge Number: 1
Start Time: 03:00
End Time: 09:15

 Last Finisher Details:
Name: Chikuu
Badge Number: 3
Start Time: 03:00
End Time: 10:00
```