17.1.1: **CharAt**

T20 IPL match is being held between the two teams “team1” and “team2”. Audience are asked to predict the winner of the match by giving a hint that the winner team has “character1” as its third character in the team name.Write a program to find the winner team.  
  
**Input Format :** Get 2 String values and a Character  
**Output Format :** Display the output String  
  
**Sample Input/Output:**  
Enter team1  
**Kolkata Knight Riders**  
Enter team2  
**Rajasthan Royals**  
Enter third character  
**j**  
Winner Team: Rajasthan Royals

**import** java.util.\*;

**import** java.lang.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Scanner in=**new** Scanner(System.in);

System.out.println("Enter team1");

String team1=in.nextLine();

System.out.println("Enter team2");

String team2=in.nextLine();

System.out.println("Enter third character");

**char** charc= in.next().charAt(0);

**char** c1=team1.charAt(2);

**char** c2=team2.charAt(2);

**if**(charc==c1)

{

System.out.println("Winner Team : "+team1);

}

**else**

{

System.out.println("Winner Team : "+team2);

}

}

}

17.2.2 **EqualsIgnoreCase**

T20 IPL board announced IPL 2016 series final match will be held at “venue1” last week. Yesterday they made another announcement that the match will be held at “venue2”. Check if both venues are same with EqualsIgnoreCase.   
  
**Sample Input/Output 1:**   
Enter venue1   
**M. chidhambaram stadium,Chennai**   
Enter venue2   
**m. Chidhambaram Stadium,Chennai**   
Both the venues are same.   
  
**Sample Input/Output 2:**   
Enter venue1   
**m. chidhambaram stadium,Chennai**   
Enter venue2   
**M. Chinnaswamy Stadium,Bangalore**   
Both the venues are different.

**Problem Requirements:**

**Java**

|  |  |  |
| --- | --- | --- |
| Keyword | Min Count | Max Count |
| equalsIgnoreCase | 1 | - |

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args)

{

Scanner in=**new** Scanner(System.in);

System.out.println("Enter team1");

String venue1=in.nextLine();

System.out.println("Enter team2");

String venue2=in.nextLine();

**if**(venue1.equalsIgnoreCase(venue2))

{

System.out.println("Both the venues are same.");

}

**else**

{

System.out.println("Both the venues are different.");

}

}

}

17.1.3: **IndexOf,LastIndexOf**

T20 IPL conducts contests for audience during the match. The contest contains 4 players and asked the audience to guess the player of the match(man of the match) . The contest also contained a clue that the first and lastIndex of character ‘a’ in the player name should be same for  player of the match.Write a program to guess the player of the match.   
[ALL Text in bold are inputs]   
  
**Sample Input/Output 1:**   
Enter the number of players   
**4**   
**Ravichandran Ashwin  
Harbhajan Singh  
Rohit Sharma  
Michael Hussey**   
Player of the Match:   
Michael Hussey   
  
**Sample Input/Output 2:**   
Enter the number of players   
**3  
Suresh Raina  
Maxwell  
Harbhajan Singh**   
Player of the Match:   
Maxwell

**Problem Requirements:**

**Java**

|  |  |  |
| --- | --- | --- |
| Keyword | Min Count | Max Count |
| lastIndexOf | 1 | - |

|  |  |  |
| --- | --- | --- |
| Keyword | Min Count | Max Count |
| indexOf | 1 | - |

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Scanner in=**new** Scanner(System.in);

System.out.println("Enter the number of players");

**int** n=in.nextInt();

in.nextLine();

String[] a=**new** String[100];

**for**(**int** i=0; i<n; i++)

{

a[i]=in.nextLine();

}

**for**(**int** i=0; i<n; i++)

{

**if**(a[i].indexOf('a')==a[i].lastIndexOf('a'))

{

System.out.println("Player of the Match:");

System.out.println(a[i]);

System.exit(0);

}

}

}

}

17.1.4: **SubSequence**

T20 IPL  generated subsequences of team names. Write a program to get a team name, starting index and ending index and generate a  
  
  
subsequence using subsequence function.

**Sample Input/Output 1:**

Enter team name   
**Delhi DareDevils**   
Enter starting index of the sequence   
**6**   
Enter ending index of the sequence   
**16**   
DareDevils

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args)

{

Scanner in=**new** Scanner(System.in);

System.out.println("Enter team name");

String name=in.nextLine();

System.out.println("Enter starting index of the sequence");

**int** a=in.nextInt();

System.out.println("Enter ending index of the sequence");

**int** b=in.nextInt();

System.out.println(name.subSequence(a,b) );

}

}

17.1.5: **SubString**

T20 IPL conducts a contest during the match to get the short name of players. Write a program to get the short name of  
  
  
players using Substring method.

**Sample Input/Output 1:**

Enter Player name   
**Virat Kohli**   
Enter starting index   
**6**   
Short name of Virat Kohli: Kohli

**import** java.util.\*;

**class** Main

{

**public** **static** **void** main(String args[])

{

Scanner in=**new** Scanner(System.***in***);

System.***out***.println("Enter player name");

String name=in.nextLine();

System.***out***.println("Enter index");

**int** n=in.nextInt();

System.***out***.println("The short name of "+ name +" is "+name.substring(n));

}

}

17.1.6: **startswith,endswith**

T20 IPL conducts a contest during the match to get the name of players which either starts with M or ends with a.  
  
  
Write a program to display the name of the players in the format as given in sample input and output.

**Sample Input and Output :**  
  
  
  
Enter the number of players  
  
**5**  
  
Enter the player name  
  
**Mahendra Singh Dhoni    
  
Michael Hussey  
  
Rohit Sharma  
  
Rahul Sharma  
  
Ravichandran Ashwin**  
  
Player name starting with 'M' or Ending with 'a'  
  
Mahendra Singh Dhoni  
  
Michael Hussey  
  
Rohit Sharma  
  
Rahul Sharma

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Scanner in=**new** Scanner(System.***in***);

System.***out***.println("Enter the number of players");

**int** n=in.nextInt();

System.***out***.println("Enter the player name");

String a[]=**new** String[100];

in.nextLine();

**for**(**int** i=0; i<n; i++)

{

a[i]=in.nextLine();

}

System.***out***.println("Player name starting with 'M' or Ending with 'a'");

**for**(**int** i=0; i<n; i++)

{

**if**(a[i].startsWith("M")||a[i].endsWith("a"))

{

System.***out***.println(a[i]);

}

}

}

}

17.1.7: **RegionMatches**

T20 IPL board does a survey to find how many players have same Michael as their first names. Write a program to get two player names and check if their firstnames is Michael using regionmatches function.   
  
**Input and Output Format:**   
Refer sample input and output for formatting specifications.   
  
**Sample Input/Output 1:**   
Enter player names   
**Michael Lumb  
Michael Clarke**   
Both the players names starts with Michael   
  
**Sample Input/Output 2:**   
Enter player names   
**Michael Lumb  
Mitchell Johnson**   
Both the players names does starts with Michael

**Problem Requirements:**

**Java**

|  |  |  |
| --- | --- | --- |
| Keyword | Min Count | Max Count |
| regionMatches | 1 | - |

**import** java.util.\*;

**class** Main

{

**public** **static** **void** main(String args[])

{

Scanner in=**new** Scanner(System.***in***);

**int** count=0;

System.***out***.println("Enter players name");

String a[]=**new** String[3];

//in.nextLine();

**for**(**int** i=0;i<2;i++)

{

a[i]=in.nextLine();

}

**for**(**int** i=0;i<2;i++)

{

**if**(a[i].regionMatches(0,"Michael",0,7))

{

count++;

}}

**if**(count==2)

{

System.***out***.println("Both the players names starts with Michael");

}

**else**

{

System.***out***.println("Both the players names does not starts with Michael");

}

}

}

17.1.8: **Replace**

T20 IPL gets the team details as follows,   
Chennai Super Kings – Captain – Dhoni   
A captain is also called as Skipper .Write a program to replace the word captain with skipper   
  
**Input and Output Format:**   
Refer sample input and output for formatting specifications.   
  
**Sample Input/Output 1:**   
Enter team details   
**Royal Challengers Bangalore – Captain – Virat Kohli**   
After replacement   
Royal Challengers Bangalore – Skipper – Virat Kohli

**Problem Requirements:**

**Java**

|  |  |  |
| --- | --- | --- |
| Keyword | Min Count | Max Count |
| replaceAll | 1 | - |

**import** java.util.\*;

**class** Main

{

**public** **static** **void** main(String args[])

{

Scanner in=**new** Scanner(System.***in***);

**int** count=0;

System.***out***.println("Enter team details");

String s1=in.nextLine();

//in.nextLine();

String s2=s1.replaceAll("Captain", "Skipper");

System.***out***.println(s2);

}

}

17.1.9: **Contains**

T20 IPL board get number of player names and finds the players with their last name as “Sharma”. Write a program to get the player names in an array and use Contains method to get the players with their last name as Sharma.   
  
**Input and Output Format:**   
Refer sample input and output for formatting specifications.   
  
**Sample Input/Output 1:**   
Enter number of players   
**5**   
Enter player names   
**Rohit Sharma  
Adam Smith  
Ishant Sharma  
Mohit Sharma  
Jaspirit Bumrah**   
Rohit Sharma   
Ishant Sharma   
Mohit Sharma

**Problem Requirements:**

**Java**

|  |  |  |
| --- | --- | --- |
| Keyword | Min Count | Max Count |
| contains | 1 | - |

**import** java.util.\*;

**class** Main

{

**public** **static** **void** main(String args[])

{

Scanner in=**new** Scanner(System.***in***);

System.***out***.println("Enter number of players");

**int** n =in.nextInt();

System.***out***.println("Enter player names");

String a[]=**new** String[n];

in.nextLine();

**for**(**int** i=0;i<n;i++)

{

a[i]=in.nextLine();

}

**for**(**int** i=0;i<n;i++)

{

**if**(a[i].contains("Sharma"))

{

System.***out***.println(a[i]);

}

}

}

}

17.2.1:

Date validation

T20 IPL system stores the date of all the final matches in a particular format.   
Check whether the date is set as per the following Validation Rule :   
1. The date should be in the format dd-MM-yyyy. It is valid only when the date is in this format dd-MM-yyyy.   
Include a class UserMainCode with a static method validateDate which accepts a string. In this method check whether the given date is valid as per the validation rules mentioned above. i.e the date should be a data in the format dd-mm-yyyy. The return type is Boolean.   
  
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.   
  
**Input and Output Format:**   
Input consists of a string.   
Output consists of a string “Valid” or “Invalid”.   
Refer sample output for formatting specifications.   
  
**Sample Input 1:**   
01-06-2008   
**Sample Output 1:**   
Valid   
  
**Sample Input 2:**   
24/05/2009   
**Sample Output 2:**   
Invalid

**import** java.io.\*;

**import** java.text.\*;

//import java.time.temporal.ValueRange;

**import** java.util.Date;

**public** **class** Main {

**public** **static** **void** main(String[] args) **throws** IOException, ParseException {

// **TODO** Auto-generated method stub

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.***in***));

String s=br.readLine();

**if**(UserMainCode.*validateDate*(s))

{

System.***out***.println("Valid");

}

**else**

System.***out***.println("Invalid");

}

}

**public** **class** UserMainCode {

**public** **static** **boolean** validateDate(String string)

{

**int** flag=0;

**for**(**int** i=0;i<string.length();i++)

{

**if**(string.charAt(i)=='-')

flag++;

}

**if**(flag!=2)

**return** **false**;

**else**

{

String[] news=string.split("-");

**int** days=Integer.*parseInt*(news[0]);

**int** months=Integer.*parseInt*(news[1]);

**if**(days<=31 && months<=12)

**return** **true**;

**else**

**return** **false**;

}

}

}

17.2.2:

**Validation -- II**

In T20 IPL all the team names have more than 5 characters.So the board decided to indicate the teams by using the following format.   
Consider Chennai Super Kings   
then it is called as CSK-Chennai Super Kings   
Check whether the team name is set as per the following Validation Rule :   
1. The team name should have the first alphabet of all the words in the team name,followed by hyphen,then followed by the team name. It is valid only when the team name is in this format.   
Include a class UserMainCode with a static method validateTeam which accepts a string. In this method check whether the given team name is valid as per the validation rules mentioned above.The return type is Boolean.   
  
    
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.   
  
**Input and Output Format:**   
Input consists of a string.   
Output consists of a string “Valid” or “Invalid”.   
Refer sample output for formatting specifications.   
  
**Sample Input 1:**   
Delhi DareDevils-DD   
**Sample Output 1:**   
Valid   
  
**Sample Input 2:**   
Royal Challengers Bangalore RCB   
**Sample Output 2:**   
Invalid

**import** java.text.\*;

**import** java.time.temporal.ValueRange;

**import** java.util.Date;

**import** java.io.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) **throws** IOException, ParseException {

// **TODO** Auto-generated method stub

BufferedReader br=**new** BufferedReader(**new** InputStreamReader(System.***in***));

String team=br.readLine();

**if**(UserMainCode.*validateTeam*(team))

{

System.***out***.println("Valid");

}

**else**

System.***out***.println("Invalid");

}

}

**public** **class** UserMainCode {

**public** **static** **boolean** validateTeam(String string)

{

**int** flag=0,count=0;

**for**(**int** i=0;i<string.length();i++)

{

**if**(string.charAt(i)=='-')

flag++;

}

**if**(flag!=1)

**return** **false**;

**else**

{

String[] teamName=string.split("-");

String full\_name=teamName[0];

String short\_name=teamName[1];

String[] partsName=full\_name.split(" ");

String[] parts=**new** String[partsName.length];

**for**(**int** i=0;i<partsName.length;i++)

{

parts[i]=partsName[i];

**if**(short\_name.charAt(i)==parts[i].charAt(i))

count++;

}

**if**(count!=partsName.length)

**return** **true**;

**else**

**return** **false**;

}

}

}

17.2.3:

|  |
| --- |
| **Input and Output Format:**  **Input :** consists of three strings.  First line indicates the player name  Second line indicates the player country  Third line indicates the player format   **Output :** consists of a string “Valid” or “Invalid”.  Refer sample output for formatting specifications.   **Sample Input 1:**  Michael Hussey  Australia  Michael Hussey#AUS  **Sample Output 1:**  Valid   **Sample Input 2:**  Dale Steyn  South Africa  Dale Steyn#SOU  **Sample Output 2:**  Invalid |

Top of Form

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Scanner tom=**new** Scanner(System.***in***);

String a=tom.nextLine();

String b=tom.nextLine();

String c=tom.nextLine();

UserMainCode us=**new** UserMainCode();

**if**(us.validatePlayer(a, b, c))

{

System.***out***.println("Valid");

}

**else**

{

System.***out***.println("Invalid");

}

}}

**class** UserMainCode {

**public** **boolean** validatePlayer(String a,String b,String c)

{

String x="";

**if**(b.contains(" "))

{

x=b.substring(0, 1);

**for**(**int** i=0;i<b.length();i++)

{

**char** e=b.charAt(i);

**if**(e==' ')

{

x=x+b.substring(i+1,i+2);

x=x.toUpperCase();

}

}

}

**else**

{

x=b.substring(0,3);

}

String[] ar=c.split("#");

**if**(ar[1].equals(x))

{

**return** **true**;

}

**else**

{

**return** **false**;

}

}

}

17.2.4: **Validation -- IV**

In T20 IPL all the team names have their home ground .The city of their home ground is given in particular format.   
Check whether the city name is set as per the following Validation Rule :   
1. The city name should have only the first two characters and last two characters of the word and the remaining letters should be replaced with asterisks (‘\*’) .The city name should contain only alphabets .It is valid only when the name is in this format.   
Include a class UserMainCode with a static method validateCity which accepts a string. In this method check whether the given city name is valid as per the validation rules mentioned above.The return type is Boolean.   
  
 Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.   
  
**Input and Output Format:**   
Input consists of a string.   
Output consists of a string “Valid” or “Invalid”.   
Refer sample output for formatting specifications.   
**Sample Input 1:**   
Ch\*\*\*ai   
**Sample Output 1:**   
Valid   
  
**Sample Input 2:**   
Mum\*\*i   
**Sample Output 2:**   
Invalid

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Scanner a=**new** Scanner(System.***in***);

String b=a.nextLine();

UserMainCode.*validateCity*(b);

}

}

**class** UserMainCode

{

**public** **static** String validateCity(String b)

{

String a=**null**;

**char** []c=b.toCharArray();

**int** count=0,count1=0,count2=0;

**for**(**int** i=0;i<2;i++)

{

**if**( (c[i]>='a' && c[i]<='z') || (c[i]>='A' && c[i]<='Z'))

{

count1++;

}

}

**for**(**int** i=2;i<c.length-2;i++)

{

**if**(c[i]=='\*')

{

count++;

}

}

**for**(**int** i=c.length-2;i<c.length;i++)

{

**if**( (c[i]>='a' && c[i]<='z') || (c[i]>='A' && c[i]<='Z'))

{

count2++;

}

}

**int** g=c.length-4;

**if**(count1==2 && count2==2)

{

**if**(count==g)

{

System.***out***.println("Valid");

a= "Valid";

}

**else**

System.***out***.println("Invalid");

}

**else**{

System.***out***.println("Invalid");

a="Invalid";

}

**return** a;

}

}

17.2.5:

|  |
| --- |
| **Validation -- V**  In T20 IPL Umpires considers a over as valid ,if the over has a wicket and does not have  any no balls.  Check whether the over is set as per the following Validation Rule :​  1. The over is a string with six characters,each character is a delivery . One Over should  have one or more wickets ,represented as 'W' and should not have 'no balls' , it is  represented as 'N'.It is valid only when the over is in this format.   Include a class UserMainCode with a static method validateOver which accepts a string. In this  method check whether the given over is valid as per the validation rules mentioned above.The return  type is Boolean.​  ​  Create another class  named Main and write a Main method to test the above class and accept the string and call the static method present inUserMainCode.​  **​ Input and Output Format:​**  **Input :** consists of a string.  **Output :** consists of a string “Valid” or “Invalid”.​  Refer sample output for formatting specifications.   **Sample Input 1:**  0W001W  **Sample Output 1:**  Valid​  **​ Sample Input 2:**  01W00N  **Sample Output 2:**  Invalid  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {  Scanner tom=**new** Scanner(System.***in***);  String a=tom.nextLine();  UserMainCode us=**new** UserMainCode();  **if**(us.validateOver(a))  {  System.***out***.println("Valid");  }  **else**  {  System.***out***.println("Invalid");  }  }  }  **public** **class** UserMainCode {  **public** **boolean** validateOver(String a)  {  **int** count=0;  **if**(a.length()==6 && a.contains("W"))  {  **for**(**int** i=0;i<6;i++)  {  **char** x=a.charAt(i);  **if**(x=='0' || x=='1' || x=='2' || x=='3' || x=='4' || x=='5' || x=='6' || x=='W' || x=='w')  {  count++;  }  }  **if**(count==6)  {  **return** **true**;  }  **else**  {  **return** **false**;  }  }  **else**  {  **return** **false**;  }  }  }  Or  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {  Scanner tom=**new** Scanner(System.***in***);  String a=tom.nextLine();  UserMainCode us=**new** UserMainCode();  **if**(us.validateOver(a))  {  System.***out***.println("Valid");  }  **else**  {  System.***out***.println("Invalid");  }  }  }  **class** UserMainCode {  **public** **boolean** validateOver(String a)  {  **if**(a.length()==6)  {  **if**(a.matches("[w,W,0-6]{6}"))  **return** **true**;  **else**  **return** **false**;  }  **else**  **return** **false**;  } }  **17.2.6 Validation -- VI**    In T20 IPL Player name is displayed along with the team name and the total runs of the player in the season.  Orange Cap Details: 1. The input is a string it consists of team name followed by player name and player's score in the tournament .The team name either has three characters or third is a Space if it contain only 2 letters.Display the details by spliting team name ,player name and his score as given in sample input and output  Include a class UserMainCode with a static method **OrangeCapDetails** which accepts a string. In this method display the player details as given in sample input and output .The return type is void.   Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  **Input and Output Format:** Input consists of a string.  Refer sample output for formatting specifications.  **Sample Input 1:**  CSKMathew Hayden572  **Sample Output 1:**  CSK  Mathew Hayden  572  **Sample Input 2:**  MI Sachin Tendulkar618  **Sample Output 2:**  MI  Sachin Tendulkar  618  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args)  {  Scanner sc=**new** Scanner(System.***in***);  String str=sc.nextLine();  UserMainCode u=**new** UserMainCode();  u.*OrangeCapDetails*(str);  }  }  **public** **class** UserMainCode {  **static** **void** OrangeCapDetails(String s)  {  String str1,str2,str3;  **int** count=0;  **int** i;  str1=s.substring(0,3);  **for**( i=0;i<s.length();i++)  {  **if**(Character.*isDigit*(s.charAt(i)))  {  **break**;  }  }  str2=s.substring(3,i);  str3=s.substring(i,s.length());  **for**(i=0;i<3;i++)  {  **if**(str1.charAt(i)==' ')  {  count++;  }  }  **if**(count==1 || count==0)  {  System.***out***.println(str1+"\n"+str2+"\n"+str3);  }  }  }  17.2.7: **Validation -- VII**    In T20 IPL ,a player name is considered as lucky based on some constraints.  Check whether the player name is set as per the following Validation Rule : 1. The player name is a string ,it should have alphabet 'a' only at odd occurrences It is valid only when the player name is in this format.If alphabet 'a' is not present in the whole string also it is valid only.  2.If it present in even occurrence,the output must be invalid.  Include a class **UserMainCode** with a static method **validatePlayer** which accepts a string. In this method check whether the given player name is valid as per the validation rules mentioned above.The return type is Boolean.   Create a Class **Main** which would be used to accept the string and call the static method present in UserMainCode. **Input and Output Format:** Input consists of a string.  Output consists of a string “Valid” or “Invalid”. Refer sample output for formatting specifications.  **Sample Input 1:** Albie Morkel **Sample Output 1:** Valid **Sample Input 2:** Suresh Raina **Sample Output 2:** Invalid  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) {  Scanner tom=**new** Scanner(System.***in***);  String a=tom.nextLine();  UserMainCode us=**new** UserMainCode();  **if**(us.validatePlayer(a))  {  System.***out***.println("Valid");  }  **else**  {  System.***out***.println("Invalid");  }  }  }  **public** **class** UserMainCode {  **public** **boolean** validatePlayer(String a)  {  **int** count=1;  **for**(**int** i=0;i<a.length();i++)  {  **if**(i%2!=0)  {  **if**(a.charAt(i)=='a' || a.charAt(i)=='A')  {  count=0;  }  }  }  **if**(count==0)  {  **return** **false**;  }  **else**  {  **return** **true**;  }  }  }  17.2.8 **Validation -- VIII**    In T20 IPL ,team name has many occurrence of vowels  Include a class UserMainCode with a static method **vowelcount** which accepts a string and returns the number of occurrences of vowels .The return type is int.   Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.  **Input and Output Format:** Input consists of a string. Output consists of an integer. Consider case insensitive comparison. Refer sample output for formatting specifications.  **Sample Input 1:**  Kolkata Knight Riders  **Sample Output 1:**  6 **Sample Input 2:**  Rajasthan Royals  **Sample Output 2:**  5  **import** java.util.\*;  **public** **class** Main {  **public** **static** **void** main(String[] args) **throws** Exception{  Scanner tom=**new** Scanner(System.***in***);  **int** count=0;  String a=tom.nextLine();  String b="aeiou";  **for**(**int** i=0;i<a.length();i++)  {  **for**(**int** j=0;j<b.length();j++)  {  **if**(a.charAt(i)==b.charAt(j))  count++;  }  }  System.***out***.println(count);  tom.close();  }  } |

1Top of Form

Bottom of Form

Top of Form

Bottom of Form

Bottom of Form

|  |
| --- |
| **List 1**  The Chennai Super Kings were the most successful team in the IPL with a win percentage of 60.68 and had won the title twice in succession (2010 and 2011).    The CSKs have played “n” matches so far in IPLs. Given the number of matches “n” that CSK has played and their team score in the matches as a list, write a program to find the total runs and the average runs scored by the team in all “n” matches.    **Input Format:**  First line of the input is an integer “n” that corresponds to the number of matches played by CSK.  Next “n” lines contains an integer in each line, that corresponds to the runs scored by CSK in each of the “n” matches.    **Output Format:**  Output should print in the first line the integer that gives the total runs scored by CSK in the matches.  In the second line, print a float value that gives the average runs.    **Sample Input :**  5  200  210  180  176  192    **Sample Output :**  958  191.6 |

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String args[])

{

**int** n,sum=0;

Scanner sc=**new** Scanner(System.***in***);

n=sc.nextInt();

ArrayList<Integer> list=**new** ArrayList<Integer>();

**for**(**int** i=0;i<n;i++)

{

list.add(sc.nextInt());

sum+=list.get(i);

}

System.***out***.println(sum+"\n"+(**float**)sum/n);

}

}

2. **List 2**

The vice-captain and the leading run-scorer of the team CSK is Suresh Raina. One of the best fielders in world cricket, Raina is considered as the finest T20 batsman.

Given the number of matches “n” that Raina has played in IPL and his individual score in the matches as a list, write a program to sort the runs scored by him.

**Input Format:**

First line of the input is an integer “n” that corresponds to the number of matches played by Raina.

Next “n” lines contains an integer in each line, that corresponds to the runs scored by Raina in each of the “n” matches.

**Output Format:**

Output should print the runs scored by Raina in sorted order, in “n” lines.

**Sample Input :**

6

101

78

90

59

77

67

**Sample Output :**

59

67

77

78

90

101

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String args[])

{

**int** n;

Scanner sc=**new** Scanner(System.***in***);

n=sc.nextInt();

ArrayList<Integer> arr=**new** ArrayList<Integer>();

**for**(**int** i=0;i<n;i++)

{

arr.add(sc.nextInt());

}

Collections.*sort*(arr);

**for**(**int** i:arr)

System.***out***.println(i);

}

}

3. **List 3**

Team CSK played its home matches at the M. A. Chidambaram Stadium in Chennai. They remain unbeaten in their home matches but were a real threat to their opponents at away grounds too. Other venues where CSK had played were M. Chinnaswamy Stadium, Wankhede Stadium, Eden Gardens,Green Park Stadium,Brabourne Stadium etc.,  
  
Assume the number of matches in total played by CSK is “n” and the venues of each of these matches are given in a string list. Write a program to find the frequency of matches that CSK had played in a particular venue “x”.  
  
**Input Format:**  
First line of the input is an integer “n” that corresponds to the number of matches played by CSK.  
Next “n” lines contains each a string corresponding to the venues that CSK played in.  
Last line contains a string “x” that corresponds to the particular venue whose frequency has to be found.  
  
**Output Format:**  
Output should display an integer in a single line, that corresponds to the frequency of the matches played in a given venue “x”.  
  
**Sample Input :**  
8  
M. A. Chidambaram Stadium  
M. A. Chidambaram Stadium  
M. A. Chidambaram Stadium  
M. Chinnaswamy Stadium  
M. Chinnaswamy Stadium  
Wankhede Stadium  
Eden Gardens  
M. A. Chidambaram Stadium  
M. A. Chidambaram Stadium  
  
**Sample Output :**  
4

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])throws Exception

{

int n;

//Scanner sc=new Scanner(System.in);

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

n=Integer.parseInt(br.readLine());

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

for(int i=0;i<n;i++)

{

arr.add(br.readLine());

}

String key=br.readLine();

System.out.println(Collections.frequency(arr,key));

}

}

 4.  A measure of a batsman's greatness is his ability to score runs on foreign conditions. It is quite obvious in Cricket that most batsmen have been excellent at home grounds but flops overseas. Consequently not many teams have aggregates balanced in terms of home and away performances.

Sunil now wanted to analyze the performance of IPL teams based on the runs scored in home as well away matches. Given are the team name, number of matches played by the team in home ground “n” and away grounds “m” respectively, runs scored by the team in each of the matches both home and away respectively. Write a program to store the runs scored by the team in both home ground and in other grounds in a list and help Sunil to display the score (in both home and away grounds) of the team that is greater than 300.

**Input Format:**

First line of the input contains a string that gives the name of the IPL team.

Second line of input contains the integer “n” that corresponds to the number of matches played by the team in home grounds.

Next “n” lines contains the runs scored by the team in each of the matches in home grounds.

Next line that follows contains the integer “m” that corresponds to the number of matches played by the team in away grounds.

Next “m” lines contains the runs scored by the team in each of the matches in away grounds.

**Output Format:**

Output should display the runs scored by the team in both home ground and in other grounds as a list, line after line.

In the lines to follow, the output should display the score (in both home and away grounds) of the team that is greater than 300, line after line.

**Sample Input and output :**   
  
Enter the team name   
**Chennai Super Kings**   
Enter the number of matches played in home ground   
**2**   
Enter the runs scored   
**290  
320**   
Enter the number of matches played in other ground   
**3**   
Enter the runs scored   
**399  
180  
150**   
Runs scored by Chennai Super Kings   
290   
320   
399   
180   
150   
Run scored by Chennai Super Kings more than 300   
320   
399

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])throws Exception

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

ArrayList<Integer> rh=new ArrayList<Integer>();

ArrayList<Integer> ra=new ArrayList<Integer>();

System.out.println("Enter the team name");

String name=br.readLine();

System.out.println("Enter the number of matches played in home ground");

int mh=Integer.parseInt(br.readLine());

System.out.println("Enter the runs scored");

for(int i=0;i<mh;i++)

rh.add(Integer.parseInt(br.readLine()));

System.out.println("Enter the number of matches played in other ground");

int ma=Integer.parseInt(br.readLine());

System.out.println("Enter the runs scored");

for(int i=0;i<ma;i++)

ra.add(Integer.parseInt(br.readLine()));

System.out.println("Runs scored by "+name);

for(int i:rh)

System.out.println(i);

for(int i:ra)

System.out.println(i);

System.out.println("Run scored by "+name+" more than 300");

for(int i:rh){

if(i>300)

System.out.println(i);}

for(int i:ra){

if(i>300)

System.out.println(i);}

}

}

5. **List 5**

Cricket followers often believe that an essential characteristic of excellence is Consistency or low variability of performance. IPL has seen numerous such consistent batsmen who set the stage alight with their mastery.

There are two lists which contains the names of players who were the top 5 scorers of two seasons 8 and 9 of IPL respectively. Write a program to find those players who have shown a consistent play in both the seasons. Precisely find the players who are on the list of top scorers in both the IPL seasons. Use retailAll method.

**Input Format:**

First 5 lines of the input contains the names of players who were the top scorers of IPL season 8.

Second 5 lines of the input contains the names of players who were the top scorers of IPL season 9.

**Output Format:**

Output should print the names of common players in both seasons line after line.

**Sample Input and output:**

Enter the top 5 scorers of IPL Season 8

**David Warner**

**AM Rahane**

**LMP Simmons**

**AB de Villiers**

**Virat Kohli**

Enter the top 5 scorers of IPL Season 9

**Virat Kohli**

**David Warner**

**AB de Villiers**

**Shikhar Dhawan**

**Gautam Gambhir**

Consistent run scorers

David Warner

AB de Villiers

Virat Kohli

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])throws Exception

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the top 5 scorers of IPL Season 8");

ArrayList<String> i8=new ArrayList<String>();

for(int i=0;i<5;i++)

i8.add(br.readLine());

System.out.println("Enter the top 5 scorers of IPL Season 9");

ArrayList<String> i9=new ArrayList<String>();

for(int i=0;i<5;i++)

i9.add(br.readLine());

System.out.println("Consistent run scorers");

for(int i=0;i<5;i++)

for(int j=0;j<5;j++)

if(i8.get(i).equals(i9.get(j)))

System.out.println(i8.get(i));

}

}

6. **List 6**

IPL matches has a huge fan base than any other famous leagues in the world. Sunil, such an ardent fan of IPL, follows the happenings of the league very closely and continues to cheer his favourite players and the team they belong to, year after year.

There is a list containing the team rankings of the top 5 teams of IPL season. Sunil wished to know the team which was in the nth position in the ranking. Given the input as “n”, write a program to get the name of the team in the nth position. Use get method.

**Input Format:**

First 5 lines of the input contains the names of top 5 teams of IPL season.

Sixth line is an integer that corresponds to the position in the ranking table that Sunil wished to know the team name for.

**Output Format:**

Output should display in a single line, the name of the IPL team which was in the nth position of the ranking.

**Sample Input/Output:**

Enter the teams in ranking table

**Sunrisers Hyderabad**

**Gujarat Lions**

**Royal Challengers Bangalore**

**Kolkata Knight Riders**

**Mumbai Indians**

Enter the rank to be searched

**3**

Royal Challengers Bangalore

import java.util.\*;

import java.io.\*;

public class Main {

public static void main(String args[])throws Exception

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the teams in ranking table");

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

for(int i=0;i<5;i++)

name.add(br.readLine());

System.out.println("Enter the rank to be searched");

int index=Integer.parseInt(br.readLine());

System.out.println(name.get(index-1));

}

}

7. **List 7**

With a news of the brand new season of IPL 10 announced, it promises to be yet another cricketing extravaganza. Defending champions of the title for IPL 10 are team Sunrisers Hyderabad. The same 8 teams of season 9 are going to compete and we may be in for a humdinger of a season which sees a tough fight for the top 4 spots.

There is a list which contains the team rankings of the top 5 teams of IPL season 9. It is predicted by statistcians that these same 5 teams would sure retain in top 5 positions of IPL 10 as well but in different positions. Write a program that prints the prediction of rankings of those 5 teams that is obtained by swapping two given positions by the user. Use swap method.

**Input Format:**

First 5 lines of the input contains the names of top 5 teams of IPL season 9.

Sixth line of the input contains an integer that corresponds to the first swap position.

Seventh line of the input contains an integer that corresponds to the second swap position.  
  
  
**Sample Input and Output :**  
  
**Sunrisers Hyderabad  
Gujarat Lions  
Royal Challengers Bangalore  
Kolkata Knight Riders  
Mumbai Indians**  
Enter swap positons  
 **2**  
Royal Challengers Bangalore  
Gujarat Lions  
Sunrisers Hyderabad  
Kolkata Knight Riders  
Mumbai Indians

import java.util.\*;

import java.io.\*;

class Main

{

public static void main(String args[])throws Exception

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

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

int i;

for( i=0;i<5;i++)

{

l.add(br.readLine());

}

System.out.println("Enter swap positons");

int a= Integer.parseInt(br.readLine());

int b= Integer.parseInt(br.readLine());

Collections.swap(l,a,b);

for(int j=0;j<5;j++)

{

System.out.println(l.get(j));

}

}}

**List 8**

Cricket is a sport that generates a large number of statistics. Statistics are recorded for each player during a match, and aggregated over a career. At the professional level, statistics for Test cricket, one-day internationals, and Twenty20 limited over matches are recorded separately.   
  
Sunil now aims to record the statistics of a player to be displayed during the IPL matches as a list. Since he is a beginner in the task, he may not be precise. He gets the details of a player like Name, Age, Country firstly from the user. Later he realized he has missed the statistics of the player's Skill and also the position of that detail in the list. He gets these two inputs thereafter and now wants to remove a specific detail from the list. Help him with the usage of the set() and remove() methods to facilitate his task and display the desired output.   
  
**Input Format:**   
First line of the input is a string that corresponds to the Name of the player.   
Second line of the input is an integer that corresponds to the Age of the player.   
Third line of the input is a string that corresponds to the Country where the player belongs.   
(Next three lines should print these 3 details of the player).   
Fourth line of the input contains a string that corresponds to the Skill of the player. It might be one of these options – Batsman/Bowler/All-rounder/Wicket Keeper   
  
Fith line of the input is an integer that corresponds to the position in the list where the player detail 'Skill' has to be inserted.   
(Next four lines should print the 4 details name, age, country and skill of the player).   
Sixth line of the input is an integer that corresponds to the position of detail that is to be removed in the list.   
(Next three lines should print the remaining 3 details of the player excluding the removed detail).   
  
**Output Format:**   
Output should display the first 3 details name, age and country of the player, line after line.   
After adding the 4th detail skill of the player, output should display the 4 details name, age and country and skill of the player, line after line.   
After removing any desired detail of the player from the list, should print the remaining 3 details of the player excluding the removed detail.   
**[All text in bold corresponds to input.]  
  
Sample Input and Output :**   
Enter the player details   
Enter player name   
**Dhoni**   
Enter age   
**35**   
Enter Country   
**India**   
Player Details   
Dhoni   
35   
India   
Enter Skill   
**All Rounder**   
Enter the position to add the skill   
**2**   
Player Details   
Dhoni   
35   
All Rounder   
Enter the position of the detail to be removed   
**1**   
Player Details   
Dhoni   
All Rounder

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.Collections;

import java.util.List;

public class Main {

public static void main(String args[]) throws Exception, IOException

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

List<String> a=new ArrayList<String>();

System.out.println("Enter the player details");

System.out.println("Enter player name");

a.add(br.readLine());

System.out.println("Enter age");

a.add(br.readLine());

System.out.println("Enter Country");

a.add(br.readLine());

System.out.println("Player Details");

for(String b:a)

{

System.out.println(b);

}

System.out.println("Enter Skill");

String skill=br.readLine();

System.out.println("Enter the position to add the skill");

int n=Integer.parseInt(br.readLine());

a.remove(n);

a.add(n,skill);

System.out.println("Player Details");

for(String b:a)

{

System.out.println(b);

}

System.out.println("Enter the position of the detail to be removed");

int m=Integer.parseInt(br.readLine());

a.remove(m);

System.out.println("Player Details");

for(String b:a)

{

System.out.println(b);

}

}

}

9. **IPL PlayerList**

2015 was the last time CSK played the IPL. In the season, the team's owner finalised “n” players in its squad at the auction including oversees players like Mike Hussey, Dwayne Smith, etc., But due to the fact that few players were down due to injuries during the warm up matches, the selection committee had to remove few players from the squad and add few other players.   
  
The names of “n” players are stored in a list. Write a program to insert/delete players names' from the list and also to print the final modified list.   
  
**Input Format:**   
  
First line of the input is an integer “n” that corresponds to the number of players selected.   
  
Next “n” lines contains each a string corresponding to the names of the players.   
  
**Output Format:**   
  
Output should display the final modified list of players after insertion and deletion, line by line.   
  
**Sample Input and Output:**   
**5  
Suresh Raina  
Mike Hussey  
Dwayne Bravo  
Ravichandran Aswin  
MS Dhoni**   
Menu   
1.Insert Players   
2.Delete Players   
**1**   
**Albie Morkel**   
Player details after insertion   
Suresh Raina   
Mike Hussey   
Dwayne Bravo   
Ravichandran Aswin   
MS Dhoni   
Albie Morkel   
Do you want to continue   
**Yes**   
Menu   
1.Insert Players   
2.Delete Players   
**2  
Dwayne Bravo**   
Player details after deletion   
Suresh Raina   
Mike Hussey   
Ravichandran Aswin   
MS Dhoni   
Albie Morkel   
Do you want to continue   
**No**

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.\*;

class Main

{

public static void main(String args[])throws Exception

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String d;int h=0;

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

int n=Integer.parseInt(br.readLine());

for(int i=0;i<n;i++)

{

String a=br.readLine();

as.add(a);

}

do{

System.out.println("Menu");

System.out.println("1.Insert Players");

System.out.println("2.Delete Players");

int b=Integer.parseInt(br.readLine());

switch(b)

{

case 1:

String a=br.readLine();

as.add(a);

System.out.println("Player details after insertion");

h=n+1;

for(int i=0;i<h;i++)

{

System.out.println(as.get(i));

}

break;

case 2:

String c=br.readLine();

for(int i=0;i<n;i++)

{

if(as.get(i).equals(c))

{

as.remove(i);

}

}

h=n-1;

System.out.println("Player details after deletion");

for(int i=0;i<h;i++)

{

System.out.println(as.get(i));

}

break;

}

System.out.println("Do you want to continue");

d=br.readLine();

}while(d.equalsIgnoreCase("Yes"));

}

}

10. **Sum Scores in Even Position**

The Orange Cap will be awarded to the batsman who score most run in the IPL tournament. There are around 23+ players in each team’s squad and more than 50+ batsman of IPL who compete to grab this reward.

Assume there are “n” batsmen and their total scores put together in all the matches they played so far are given in a list. For a simple statistics it is required to know the sum of all scores of batsmen who are in the even position of the list. Write a program to find the sum of all scores in the even position of the list.

**Input Format:**

First line of the input is an integer “n” that corresponds to the number of batsmen.

Next “n” lines contains an integer which corresponds to the scores of the batsmen each.

**Output Format:**

Output should display the sum of all scores in the even position of the list as an integer in a single line.

**Sample Input**

6

765

879

779

745

898

645

**Sample Output**

2269

import java.io.\*;

import java.util.\*;

class Main

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();int sum=0;

ArrayList<Integer> as=new ArrayList<Integer>();

for(int i=0;i<n;i++)

{

as.add(sc.nextInt());

if(((i+1)%2)==0)

{

sum=sum+as.get(i);

}

}

System.out.println(sum);

sc.close();

}

}

11. **Count of 50s and 100s**

Chris Gayle, at his best, is devastating and is all about power and brute force. He is the first player to have scored a century in international Twenty20 cricket; the only man to score more than 15 centuries in the Twenty20 format; and is the leading six-hitter in Twenty20s.   
  
Assume Gayle had played “n” matches in his Twenty20 career. Given a list with “n” elements containing the scores of Gayle in the “n” matches he had played. Write a program to find the number of 50's and 100's that Gayle had scored in all “n” matches.   
  
**Input Format:**   
First line of the input is an integer “n” that corresponds to the number of matches Gayle had played.   
Next “n” lines contains an integer which corresponds to the score of Gayle in each of the matches.   
  
**Output Format:**   
Output should display an integer in the first line that gives the number of 50's Gayle has scored.   
In the second line print the integer that corresponds to the number of 100's Gayle has scored.   
  
**Sample Input**   
6   
58   
100   
49   
50   
110   
60   
**Sample Output**

3   
2 Top of Form

Bottom of Form

import java.util.\*;

import java.io.\*;

class Main

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int count=0;

int count1=0;

ArrayList<Integer> al=new ArrayList<Integer>();

for(int i=0;i<n;i++)

{

int a=sc.nextInt();

al.add(a);

if((al.get(i)>=50)&&(al.get(i)<100))

{

count++;

}

if(al.get(i)>=100)

{

count1++;

}

}

System.out.println(count);

System.out.println(count1);

}

}

12. **Team Name**

Dancing cheergirls, massive sixes, stumps splitting apart, aggression, song and dance, festivities, and a horde of TV commercial are all beckoning, when it is IPL season in the country. IPL is a rich blend of melodrama, glamour and cricket to come our way.   
  
IPL frenzied fans adorned with colourful jerseys of their favourite teams would be buzzing in the Cities for the matches to beat the living daylight out of them. Given an IPL team's name as a string “S”, write a program to help the fans print string “S” as an array list that contains each character of the string in backwards order.   
  
**Input Format:**   
First line of the input is a string “S” that corresponds to the name of the IPL team.   
  
**Output Format:**   
Output should display the string S as a list, line by line with each character of the string in backwards order.   
  
**Sample Input :**   
Gujarat Lions   
  
**Sample Output :**   
s   
n   
o   
i   
L   
    
t   
a   
r   
a   
j   
u   
G

**import** java.util.ArrayList;

**import** java.util.ListIterator;

**import** java.util.Scanner;

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

{

Scanner ip=**new** Scanner(System.***in***);

ArrayList<Character> al=**new** ArrayList<Character>();

String a=ip.nextLine();

**for**(**int** i=0;i<a.length();i++)

{

al.add(a.charAt(i));

}

ListIterator<Character> itr=al.listIterator();

**while**(itr.hasNext())

{

itr.next();

}

**while**(itr.hasPrevious())

{

System.***out***.println(itr.previous());

}

}

}

13. **Odd-Even Index**

      Over the years the quality of cricket has only gone up. Most of the star players though being an enigma to their own country teams, becomes a superstar every time the IPL starts.

There is a list containing the top “n” scores of such superstar batsmen in IPL. Rahul, a statistician has to himself a wealth of Cricket knowledge. He now intended to analyse a specific statistics with these top scores. He wanted to find and display the sum of the scores in the array list that are odd and have an even index, or the scores that are even and have an odd index or display both the sums if applicable. Write a program to help him accomplish the task.

**Input Format:**

First line of the input is an integer “n” that corresponds to the number of top scores of the batsmen.

Next “n” lines of integers contains those top scores each.

**Output Format:**

Output should display an integer in a single line, that gives the of the sum of scores in the array list that are odd and have an even index, or sum of scores that are even and have an odd index.

**Sample Input**

5

51

78

120

21

46

**Sample Output**

187

import java.util.ArrayList;

import java.util.Scanner;

public class Main {

public static void main(String args[])

{

Scanner ip=new Scanner(System.in);

ArrayList<Integer> al=new ArrayList<Integer>();

int n=ip.nextInt(),sum=0;

for(int i=0;i<n;i++)

{

int a=ip.nextInt();

al.add(a);

}

for(int i=0;i<n;i++)

{

if(((i+1)%2!=0 && al.get(i)%2==0)||((i+1)%2==0 && al.get(i)%2!=0))

sum=sum+al.get(i);

}

System.out.println(sum);

}

}

14. **Sum of Runs**

Rahul, the statistician now aims to find another statistics with the same array list of the top scores of the superstar batsmen. List contains “n” top scores of batsmen. Rahul wanted to return the sum of the scores in the array list, but consider below conditions for calculating the sum:   
  
  
--> exclude the score that is the lucky number 7   
  
--> if a score has the number 7 in it, exclude all the digits after digit 7 but consider those digits before 7. For example, if the score if 173, consider only digit “1” for calculating the sum. Leave behind digits “7” and “3”. If the score is 72, ignore the entire score.   
  
  
Write a program that determines the sum of the scores in the array list satisfying all the above conditions.   
  
  
**Input Format:**   
First line of the input is an integer “n” that corresponds to the number of top scores of the batsmen.   
  
Next “n” lines of integers contains those top scores each.   
  
  
**Output Format:**   
Output should display an integer in a single line, that gives the sum of the array satisfying all the above conditions.   
  
  
**Sample Input :**   
5   
72   
121   
171   
59   
27   
  
**Sample Output :**   
21

**import** java.util.ArrayList;

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String args[])

{

Scanner ip=**new** Scanner(System.***in***);

ArrayList<Integer> al=**new** ArrayList<Integer>();

**int** n=ip.nextInt(),sum=0,temp=0,x,r;

**for**(**int** i=0;i<n;i++)

{

**int** a=ip.nextInt();

al.add(a);

}

**for**(**int** i=0;i<n;i++)

{

x=al.get(i);

**while**(x!=0)

{

r=x%10;

sum=sum+r;

**if**(r==7)

{

sum=0;

x=x/10;

**while**(x!=0)

{

r=x%10;

sum=sum+r;

x=x/10;

}

**break**;

}

x=x/10;

}

temp=temp+sum;

sum=0;

}

System.***out***.println(temp);

}

}

15. **Count Prime Scores**

Dev, the statistician now aims to find another statistics with the same array list of the top scores of the batsmen. List contains “n” top scores of batsmen. Dev wanted to return the count of the prime scores in the array list.

Write a program that determines the count of the prime scores in the array list.

**Input Format:**

First line of the input is an integer “n” that corresponds to the number of top scores of the batsmen.

Next “n” lines of integers contains those top scores each.

**Output Format:**

Output should display an integer in a single line, that gives the count of prime scores in the list

Enter the number of matches

**6**

Enter the runs scored by the team

**151**

**160**

**197**

**199**

**223**

**190**

Number of prime scores : 4

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.\*;

public class Main {

public static void main(String[] args)throws Exception

{

int n,ch,i,j,num,count1=0,count=0;

String s,choice;

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter the number of matches");

n=Integer.parseInt(br.readLine());

ArrayList<Integer>al=new ArrayList<Integer>(n);

System.out.println("Enter the runs scored by the team");

for(i=0;i<n;i++)

{

al.add(Integer.parseInt(br.readLine()));

}

for(i=0;i<n;i++)

{

count=0;

num=al.get(i);

for(j=1;j<num/2;j++)

{

if(num%j==0)

{

count++;

}

}

if(count==1)

{

count1++;

}

}

System.out.println("Number of prime scores : "+count1);

}

}

16. **Duck - batsman's dismissal**

A duck is a batsman's dismissal for a score of zero. There are variations of this term duck as players who are dismissed by the first ball they face are said to have been dismissed for a golden duck. An opening batsman who is dismissed on the first ball of a team's innings is said to be out for a diamond duck.   
  
  
There is a string array with “n” elements that stores the name and scores of players in both the innings of a test match, in the form Name-Innings1-Innings2. Rahul now wants to find the name of the player who has scored a duck in both the innings. Write a program to display the name of that player.   
  
**Input Format:**   
  
First line of the input is an integer that “n” that corresponds to the number of elements in the arraylist.   
  
Next “n” lines contains the the name and scores of players in both the innings of a test match, in the form Name-Innings1-Innings2 as a string .   
  
  
**Output Format:**   
  
Output should display strings line by line, that gives the name of the players who has scored a duck in both the innings   
  
  
**Sample Input1 :**   
  
4   
Rohit Sharma-0-0   
Virat Kohli-0-21   
MS Dhoni-61-45   
Shan Marsh-0-0   
  
  
**Sample Output1:**   
  
Rohit Sharma   
Shan Marsh   
  
  
**Sample Input2 :**   
  
4   
Rohit Sharma-1-0   
Virat Kohli-0-21   
MS Dhoni-61-45   
Shan Marsh-0-1   
  
**Sample Output2:**   
  
No player has scored a duck

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.util.\*;

public class Main {

public static void main(String[] args)throws Exception

{

int n,i,j,count=0;

String s,choice;

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

n=Integer.parseInt(br.readLine());

ArrayList<String>al=new ArrayList<String>(n);

for(i=0;i<n;i++)

{

al.add(br.readLine());

}

for(i=0;i<al.size();i++)

{

s=al.get(i);

String[]a=s.split("-");

if(a[1].equals("0")&&(a[2].equals("0")))

{

count=1;

System.out.println(a[0]);

}

}

if(count==0)

{

System.out.println("No player has scored a duck");

}

}

}

Hashmap:

**Comparator - Team name and Number of matches**

Write a Java program to get the team name and number of matches played by the team from the user and display a report with team name and number of matches sorted based on the number of batches in ascending order. Use **Collection.sort()** method to perform the sorting in your main class. Send the Comparator object as second argument to the sort method to use this comparator for sorting.   
  
Create a main class " **Main.java**"   
Create **Team** class with below attributes,   
name - String   
numberOfMatches - Long   
Include a constructor accepting Team name and number of matches as arguments   
Add appropriate getter and setter methods for Team class   
  
Create **TeamComparator** implementing Comparator interface   
Implement compare method to compare two team objects based on their number of matches played.   
  
**Input and Output Format:**   
First input corresponds to the number of teams and followed by each team information.   
Refer sample input and output for formatting specifications.   
  
**[All text in bold corresponds to input and the rest corresponds to output]  
Sample Input/Output :**   
Enter number of teams:   
**3**   
Enter team 1 detail   
Enter Name   
**Chennai super Kings**   
Enter number of matches   
**132**   
Enter team 2 detail   
Enter Name   
**Royal Challengers Bangalore**   
Enter number of matches   
**139**   
Enter team 3 detail   
Enter Name   
**Delhi Daredevils**   
Enter number of matches   
**131**   
Team list after sort by number of matches   
Delhi Daredevils – 131   
Chennai super Kings – 132

**import** java.util.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**int** no;

Scanner sc=**new** Scanner(System.***in***);

List<Team>It=**new** ArrayList<Team>();

System.***out***.println("Enter number of teams:");

no=Integer.*parseInt*(sc.nextLine());

**for**(**int** i=0;i<no;i++)

{

System.***out***.println("Enter team "+(i+1)+" detail\nEnter Name");

String n=sc.nextLine();

System.***out***.println("Enter number of matches");

**int** nu=Integer.*parseInt*(sc.nextLine());

Team t=**new** Team(n,nu);

It.add(t);

}

Collections.*sort*(It,**new** TeamComparator());

System.***out***.println("Team list after sort by number of matches");

**for**(Team t:It)

{

System.***out***.println(t);

}

sc.close();

}

}

**public** **class** Team {

String name;

**long** numberOfMatches;

**public** Team(String name,**long** numberOfMatches)

{

**this**.name=name;

**this**.numberOfMatches=numberOfMatches;

}

**public** String getName()

{

**return** name;

}

**public** **void** setName(String name)

{

**this**.name=name;

}

**public** **long** getNumberOfMatches()

{

**return** numberOfMatches;

}

**public** **void** setNumberOfMatches(**long** numberOfMatches)

{

**this**.numberOfMatches=numberOfMatches;

}

**public** String toString()

{

**return** name+"-"+numberOfMatches;

}

}

**import** java.util.Comparator;

**public** **class** TeamComparator **implements** Comparator<Team>{

**public** **int** compare(Team a0, Team a1)

{

**return** (**int**)a0.getNumberOfMatches()-(**int**)a1.getNumberOfMatches();

}

}

1 **Comparator - Player List Based on Name and Skill**

Write a Java program to get all player details from the user and display a player list based on the skill and name.  Display a menu to select the skill of the player as shown in the sample input and output.   
Player list should be in such a way that All Rounders should be listed first, then the batsmen and then the bowlers should be displayed.Also the players should be listed in alphabetical order in each skill category.   
  
Create a main class "Main.java"   
Create Player class with below attributes,   
name - String   
skill - String   
Add appropriate getter and setter methods for Player class   
Create constructor for Player class with arguments name and skill.   
Create PlayerComparator class implementing Comparator and implementing the below method,   
public int compare(Player player1, Player player2);   
The compare method compares the two player object based on the skill and name. This method retruns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.   
  
Read all the inputs in the Main class and store the list of players in ArrayList. Use Collections.sort() method to sort the list and pass the custom comparator.   
  
  
Input and Output Format:   
First input corresponds to the number of players and followed by the details of the players.   
Refer sample input and output for formatting specifications.   
  
[All text in bold corresponds to input and the rest corresponds to output]   
Sample Input/Output :   
Please provide the number of players to be registered   
3   
Please enter player name   
Virat Kohli   
Please select the skill of the player   
1.All Rounder   
2.Batsman   
3.Bowler   
2   
Please enter player name   
MS Dhoni   
Please select the skill of the player   
1.All Rounder   
2.Batsman   
3.Bowler   
1   
Please enter player name   
Ashwin   
Please select the skill of the player   
1.All Rounder   
2.Batsman   
3.Bowler   
3   
Player Details   
Player : Ashwin Skill  : All Rounder   
Player : Virat Kohli Skill  : Batsman   
Player : MS Dhoni Skill  : Bowler

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

**int** no;

Scanner sc=**new** Scanner(System.***in***);

List<Player> lt=**new** ArrayList<Player>();

System.***out***.println("Please provide the number of players to be registered");

no=Integer.*parseInt*(sc.nextLine());

**for**(**int** i=0;i<no;i++)

{

System.***out***.println("Please enter player name");

String n=sc.nextLine();

System.***out***.println("Please select the skill of the player\n1.All Rounder\n2.Batsman\n3.Bowler");

**int** nu=Integer.*parseInt*(sc.nextLine());

**switch**(nu)

{

**case** 1:

Player t1=**new** Player(n,"All Rounder");

lt.add(t1);

**break**;

**case** 2:

Player t2=**new** Player(n,"Batsman");

lt.add(t2);

**break**;

**case** 3:

Player t3=**new** Player(n,"Bowler");

lt.add(t3);

**break**;

**default**:

System.***out***.println("Invalid Input");

**break**;

}

}

Collections.*sort*(lt,**new** PlayerComparator());

System.***out***.println("Player Details");

**for**(Player t:lt)

{

System.***out***.println(t);

}

sc.close();

}

}

**public** **class** Player {

String name;

String skill;

**public** Player(String name, String skill) {

**this**.name = name;

**this**.skill = skill;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getSkill() {

**return** skill;

}

**public** **void** setSkill(String skill) {

**this**.skill = skill;

}

**public** String toString()

{

**return** "Player : "+name+" Skill : "+skill;

}

}

**import** java.util.Comparator;

**public** **class** PlayerComparator **implements** Comparator<Player> {

@Override

**public** **int** compare(Player a0, Player a1) {

**int** c=a0.getSkill().compareTo(a1.getSkill());

**if**(c==0)

**return** a0.getName().compareTo(a1.getName());

**return** c;

}

}

**3.HashMap – Player**

Write a Java program to register all the player details [Bowler] and provide a quick view of number of wickets taken by the player based on the name of the player. Use HashMap to store all the wickets, Key will be the name of the player and value contains the player object. Multiple complaints for a user is provided using a delimiter pipe "|".

Create a main class "Main.java"

Create Player class with below attributes,

bowlerName - String

wicketCount - Integer

Add appropriate getter and setter methods for Player class

Create constructor for Player class with arguments name and complaint count

Provide search option to search based on the name of the bowler and display the number of wickets taken by him.

**Input and Output Format:**

Refer sample input and output for formatting specifications.

[All text in bold corresponds to input and the rest corresponds to output]

**Sample Input/Output :**

Enter the player name

**Aswin**

Enter wickets - seperated by "|" symbol.

**Virat Kohli|Shane Watson|Ajinkya Rahane**

Do you want to add another player (yes/no)

**yes**

Enter the player name

**Bravo**

Enter wickets - seperated by "|" symbol.

**Robin Uthapa**

Do you want to add another player (yes/no)

**no**   
Enter the player name to search

**Morkel**

No player found with the name Morkel

Do you want to search another player (yes/no)

**yes**

Enter the player name to search

**Aswin**

Player name : Aswin

Wicket Count : 3

Do you want to search another  player (yes/no)

**no**

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

String n="\"|\"";

String ans,ans1;

int f=0;

Scanner sc=new Scanner(System.in);

HashMap<String,Integer> hm=new HashMap<String,Integer>();

do{

System.out.println("Enter the player name");

String name=sc.nextLine();

System.out.println("Enter wickets - seperated by "+n+" symbol.");

String wik=sc.nextLine();

String[] tmp=wik.split("\\|");

int cnt=tmp.length;

hm.put(name, cnt);

Player p=new Player(name,cnt);

System.out.println("Do you want to add another player (yes/no)");

ans=sc.nextLine();

}

while(ans.equals("yes"));

do{

System.out.println("Enter the player name to search");

String nm=sc.nextLine();

for(Map.Entry<String,Integer> m:hm.entrySet())

{

if(nm.equals(m.getKey()))

{

System.out.println("Player name : "+m.getKey()+"\nWicket Count : "+m.getValue());

f=1;

}

}

if(f==0)

{

System.out.println("No player found with the name "+nm);

}

System.out.println("Do you want to search another player (yes/no)");

ans1=sc.nextLine();

f=0;

}

while(ans1.equals("yes"));

sc.close();

}

}

**public** **class** Player {

String bowlerName;

**int** wicketCount;

**public** Player(String bowlerName,**int** wicketCount)

{

**this**.bowlerName=bowlerName;

**this**.wicketCount=wicketCount;

}

**public** String getBowlerName()

{

**return** bowlerName;

}

**public** **void** setBowlerName()

{

**this**.bowlerName=bowlerName;

}

**public** **int** getWicketCount()

{

**return** wicketCount;

}

**public** **void** setWicketCount()

{

**this**.wicketCount=wicketCount;

}

}

4. **TreeMap-Player Details**

BCCI, for the upcoming IPL season in 2017 decided to give  unique cap numbers to every player. Player capNumber is a string. The capNumber  and player details are stored in a Treemap.   
Create a class named Player with the following private attributes --- name,team and skill. Create a list of objects of Player type.   
Cap number is the key and player details is the value. Write a program to display the details of all the players stored in this TreeMap.   
Input and Output Format:   
    
Refer sample input and output for formatting specifications.   
All text in bold corresponds to input and the rest corresponds to output.   
  
**Sample Input and Output:**   
Enter the number of players   
**2**   
Enter the details of the player 1   
**57  
Jaspirit Bumrah  
Mumbai Indians  
Bowler**   
Enter the details of the player 2   
**55  
MS Dhoni  
Rising Pune Supergiants  
All Rounder**   
Player Details   
55--MS Dhoni--Rising Pune Supergiants--All Rounder   
57--Jaspirit Bumrah--Mumbai Indians--Bowler

import java.util.Map;

import java.util.Scanner;

import java.util.TreeMap;

public class Main {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);

String capNumber;

TreeMap<String,Player> tm=new TreeMap<String,Player> ();

System.out.println("Enter the number of players");

int num=Integer.parseInt(sc.nextLine());

//sc.nextLine();

Player[] p=new Player[num];

for(int i=0;i<num;i++)

{

System.out.println("Enter the details of the player "+(i+1));

capNumber=sc.nextLine();

String a=sc.nextLine();

String b=sc.nextLine();

String c=sc.nextLine();

p[i]=new Player(a,b,c);

tm.put(capNumber, p[i]);

}

//String n=tm.get(0).getName();

//System.out.println(n);

System.out.println("Player Details");

for(Map.Entry<String,Player> m:tm.entrySet()){

System.out.println(m.getKey()+"--"+m.getValue().getName()+"--"+m.getValue().getTeam()+"--"+m.getValue().getSkill());

}

sc.close();

}

}

**public** **class** Player {

**private** String name;

**private** String team;

**private** String skill;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getTeam() {

**return** team;

}

**public** **void** setTeam(String team) {

**this**.team = team;

}

**public** String getSkill() {

**return** skill;

}

**public** **void** setSkill(String skill) {

**this**.skill = skill;

}

**public** Player(String name,String team,String skill)

{

**this**.name=name;

**this**.team=team;

**this**.skill=skill;

}

}

**HashMap – Wicket Details**

Write a java program to register all the player details [Bowler] and provide a quick view of number of wickets taken by the player based on the name of the player.

Use HashMap  to store all the wickets, Key will be the name of the player and value contains the player object. Multiple wickets for a player is provided using a delimiter pipe "|".

Create a main class "Main.java"

Create **Bowler**  class with below attributes,

name - String

Add appropriate getter and setter methods for Bowler class

Create constructor for Bowler class with argument name

Create **Wicket** class with below attributes,

playerName - String

Bowler - Bowler object

Add appropriate getter and setter methods for Wicket class

Create constructor for Wicket class with arguments description and Bowler object

Provide search option to search based on the name of the bowler and display all the wickets taken by him.

**Input and Output Format:**

Refer sample input and output for formatting specifications.

[All text in bold corresponds to input and the rest corresponds to output]

**Sample Input/Output :**

Enter the player name

**Ashwin**

Enter wickets - seperated by "|" symbol.

**Virat Kohli|Shane Watson|Ajinkya Rahane**

Do you want to add another player (yes/no)

**yes**

Enter the player name

**Bravo**

Enter wickets - seperated by "|" symbol.

**Robin Uthapa**

Do you want to add another player (yes/no)

**no**

Enter the player name to search

**Morkel**

No player found with the name Morkel

Do you want to search another player (yes/no)

**yes**

Enter the player name to search

**Ashwin**

Player Name : Ashwin

Wickets :

Virat Kohli

Shane Watson

Ajinkya Rahane

Do you want to search another player (yes/no)

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

import java.util.StringTokenizer;

public class Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner s = new Scanner(System.in);

String choice="";

HashMap<String, String> hashmap = new HashMap<>();

do{

System.out.println("Enter the player name");

String name = s.nextLine();

Bowler bow = new Bowler(name);

System.out.println("Enter wickets - seperated by \"|\" symbol");

String wickets = s.nextLine();

hashmap.put(name, wickets);

System.out.println("Do you want to add another player (yes/no)");

choice = s.nextLine();

}while(choice.equalsIgnoreCase("yes"));

int flag=0;

do{

System.out.println("Enter the player name to search");

String var = s.nextLine();

for(Map.Entry<String,String> temp : hashmap.entrySet()){

if(temp.getKey().equals(var)){

flag=1;

System.out.println("Player Name : "+temp.getKey());

System.out.println("Wickets :");

StringTokenizer st = new StringTokenizer(temp.getValue(),"|");

while(st.hasMoreTokens())

System.out.println(st.nextToken());

}

}

if(flag==0)

System.out.println("No player found with the name "+var);

System.out.println("Do you want to search another player (yes/no)");

choice = s.nextLine();

}while(choice.equalsIgnoreCase("yes"));

}

}

**public** **class** Wickets {

String playerName;

Bowler bowler;

**public** Wickets(String playerName, Bowler bowler) {

**super**();

**this**.playerName = playerName;

**this**.bowler = bowler;

}

**public** String getPlayerName() {

**return** playerName;

}

**public** **void** setPlayerName(String playerName) {

**this**.playerName = playerName;

}

**public** Bowler getBowler() {

**return** bowler;

}

**public** **void** setBowler(Bowler bowler) {

**this**.bowler = bowler;

}

}

**public** **class** Bowler {

String name;

**public** Bowler(String name) {

**super**();

**this**.name = name;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

}

3. **HashMap - Scores to Bin**

Write a Java program to group the set of scores into corresponding bin. The number of bins are fixed as 10,20,30,40. Input consists of set of scores with in a range of 1 - 40. Classify each score and increment the corresponding bin.   
Maintain the bin's in HashMap with key as 10,20,30,40 and their corresponding integer count as value.   
  
Create a main class "Main.java"   
First input corresponds to the number of overs and followed by N number of scores in each over.   
  
Create a class **Histogram** with below specification,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Access Modifier** | **Attribute/Method** | **Type/Return type** | **Name** | **Description** |
| private | Attribute | HashMap<Integer,Integer> | bins | Maintains the 4 bins and their corresponding count |
| public | Method | void | addScore(Integer) | Add the input score to corresponding bin |
| public | Method | void | displayHistogram() | Display the histogram represented by number of stars for each bin |

**Refer the sample input/output**   
  
Read the input numbers in the main class and call addScore() method in the Histogram class to add the scores to the corresponding bin.   
  
**Input and Output Format:**   
First input corresponds to the number of overs (N). Followed by N scores in each over.   
Refer sample input and output for formatting specifications.   
  
**[All text in bold corresponds to input and the rest corresponds to output]**   
**Sample Input/Output :**   
**11  
6  
17  
21  
7  
35  
22  
4  
18  
33  
7  
31**   
Histogram   
10 : \*\*\*\*   
20 : \*\*   
30 : \*\*   
40 : \*\*\*

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Scanner s = **new** Scanner(System.***in***);

**int** n= Integer.*parseInt*(s.nextLine());

Histogram h1 = **new** Histogram();

**for**(**int** i=0;i<n;i++){

**int** temp = Integer.*parseInt*(s.nextLine());

h1.addScore(temp);

}

h1.displayHistogram();

}

}

import java.util.HashMap;

import java.util.Map;

import java.util.TreeMap;

public class Histogram {

private HashMap<Integer, Integer> bins=new HashMap<Integer,Integer>();

int c10=0,c20=0,c30=0,c40=0;

public void addScore(Integer s)

{

bins.put(10, c10);

bins.put(20, c20);

bins.put(30, c30);

bins.put(40, c40);

if(s>0 && s<=10)

{

c10++;

bins.put(10, c10);

}

if(s>10 && s<=20)

{

c20++;

bins.put(20, c20);

}

if(s>20 && s<=30)

{

c30++;

bins.put(30, c30);

}

if(s>30 && s<=40)

{

c40++;

bins.put(40, c40);

}

}

public void displayHistogram()

{

Map<Integer, Integer> map = new TreeMap<Integer, Integer>(bins);

System.out.println("Histogram");

for(Map.Entry<Integer,Integer> m:map.entrySet())

{

String stars="";

for(int i=0;i<m.getValue();i++)

{

stars=stars+"\*";

}

System.out.println(m.getKey()+" : "+stars);

}

}

}

4. **TreeMap - Letter Frequency**

Write a Java program to calculate the character frequency in a sentence. The input consist of a single sentence and the output displays a graphical chart displaying the freqency of each character by number of asterisk (\*). Display the character in the output in alphabetical order. Compute the frequence of all letters except space.   
  
Use TreeMap to store the characters and frequency since the tree map maintains the entries sorted based on their natural ordering.   
  
Create a main class "Main.java"   
Create a class **LetterSequence** and include below methods and attributes,   
Include a constructor to get the sentence as the input

|  |  |
| --- | --- |
| **Method/Attribute** | **Details** |
| public TreeMap<Character,Integer> computeFrequency() | Compute the frequency of each character in the sentence and store it in the TreeMap. Return the TreeMap after the computation. |
| public void displayLetterFrequency(TreeMap<Character,Integer> frequencyMap) | Iterate the tree map and get all the entries and print the information in a graphical view as shown the sample output |
| private String sentence | Input sentence is stored in this attribute |

**Input and Output Format:**   
Refer sample input and output for formatting specifications.   
  
**[All text in bold corresponds to input and the rest corresponds to output]**   
**Sample Input/Output :**   
Enter the input string   
**Kohli is the man of the match**   
K : \*   
a : \*\*   
c : \*   
e : \*\*   
f : \*   
h : \*\*\*\*   
i : \*\*   
l : \*   
m : \*\*   
n : \*   
o : \*\*   
s : \*   
t : \*\*\*

import java.util.Scanner;

import java.util.TreeMap;

public class Main {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

System.out.println("Enter the input string");

String sentence=sc.nextLine();

LetterSequence l=new LetterSequence(sentence);

TreeMap<Character,Integer> f=l.computeFrequency();

l.displayLetterFrequency(f);

}

}

import java.util.Map;

import java.util.TreeMap;

public class LetterSequence {

private String sentence;

public TreeMap<Character,Integer> computeFrequency()

{

sentence.replaceAll("","");

String[] s1=sentence.split(" ");

String s2="";

for(int i=0;i<s1.length;i++)

{

s2=s2+s1[i];

}

int length=s2.length();

Character[] ch=new Character[length];

for(int i=0;i<length;i++)

{

ch[i]=s2.charAt(i);

}

Integer[] cts=new Integer[length];

for(int i=0;i<length;i++)

{

int count=0;

for(int j=0;j<length;j++)

{

if(ch[i].equals(ch[j]))

count++;

}

cts[i]=count;

}

TreeMap<Character,Integer> t=new TreeMap<Character,Integer>();

for(int i=0;i<length;i++)

{

t.put(ch[i], cts[i]);

}

return t;

}

public void displayLetterFrequency(TreeMap<Character,Integer> frequencyMap)

{

for(Map.Entry<Character,Integer> m:frequencyMap.entrySet())

{

String stars="";

for(int i=0;i<m.getValue();i++)

{

stars=stars+"\*";

}

System.out.println(m.getKey()+" : "+stars);

}

}

public String getSentence()

{

return sentence;

}

public void setSentence(String sentence)

{

this.sentence=sentence;

}

public LetterSequence(String sentence)

{

super();

this.sentence=sentence;

}

}

Date:

1. **Display Date**

Given a date in the form of string, write a program to convert the given string to date .   
Include a class **UserMainCode** with a static method **displayDate** which accepts a string. In this method display the given string in date format YYYY:MM:DD. The return type is void.   
  
 Create a Class **Main** which would be used to accepts a string and call the static method present in UserMainCode.   
  
**Input and Output Format:**   
Input consists of a string.   
Output consists of Date.   
  
Refer sample output for formatting specifications.   
  
**Sample Input 1:**   
May 1, 2016   
**Sample Output 1:**   
2016-05-01   
  
**Sample Input 2:**   
March 21, 2016   
**Sample Output 2:**   
2016-03-21

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) **throws** Exception {

// **TODO** Auto-generated method stub

Scanner in=**new** Scanner(System.***in***);

String str=in.nextLine();

UserMainCode n=**new** UserMainCode();

//System.out.println(str);

n.displayDate(str);

}

}

**import** java.util.\*;

**import** java.text.\*;

**public** **class** UserMainCode {

**void** displayDate(String str) **throws** Exception

{

SimpleDateFormat sdf=**new** SimpleDateFormat("MMMM dd,yyyy");

Date date=sdf.parse(str);

SimpleDateFormat sdf1=**new** SimpleDateFormat("yyyy-MM-dd");

str=sdf1.format(date);

System.***out***.println(str);

}

}

2. **Extract Date and time**

Write a program to extract date and time from the input string which is in yyyy-MM-dd HH:mm:ss date format.

Include a class UserMainCode with a static method **displayDateTime** which accepts a string. In this method display date and time in the format as given in sample input and output . The return type is void.  
  
 Create a Class Main which would be used to accept string and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of a string.  
Output should be in date format

Refer sample output for formatting specifications.

**Sample Input :**

**Enter String in this format(YYYY-MM-DD HH:mm:ss)**

2016-07-14 09:00:02

**Sample Output :**

07/14/2016, 9:00:02

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) **throws** Exception{

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("Enter String in this format(YYYY-MM-DD HH:mm:ss)");

String str=sc.nextLine();

UserMainCode umc=**new** UserMainCode();

umc.displayDateTime(str);

}

}

**import** java.text.\*;

**import** java.util.\*;

**public** **class** UserMainCode {

**void** displayDateTime(String str) **throws** Exception{

SimpleDateFormat sdf =**new** SimpleDateFormat( "yyyy-MM-dd HH:mm:ss");

Date date=sdf.parse(str);

SimpleDateFormat sdf1 =**new** SimpleDateFormat( "MM/dd/yyyy, HH:mm:ss");

str=sdf1.format(date);

System.***out***.println(str);

}

}

3. **Day Of The Year**

Given a date, write a program to display day of the year.   
Include a class **UserMainCode** with a static method **displayDay** which accepts a date. In this   
method, display day in the format as given in sample input and output . The return type is void.​   
​   
 Create a Class **Main** which would be used to accept date and call the static method present in   
UserMainCode.​   
​   
**Input and Output Format:​**   
Input consists of a string.   
Refer sample output for formatting specifications.   
  
**Sample Input :**   
2013-03-23   
**Sample Output :**   
Day of year: 82

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) **throws** Exception {

Scanner sc=**new** Scanner(System.***in***);

String str=sc.nextLine();

UserMainCode umc=**new** UserMainCode();

umc.displayDay(str);

}

}

**import** java.util.\*;

**import** java.text.\*;

**public** **class** UserMainCode {

**void** displayDay(String str)**throws** Exception{

SimpleDateFormat sdf=**new** SimpleDateFormat("yyyy-MM-dd");

Date date1=sdf.parse(str);

str=sdf.format(date1);

String word[]=str.split("-");

**int** word0=Integer.*parseInt*(word[0]);

**int** word1=Integer.*parseInt*(word[1]);

**int** word2=Integer.*parseInt*(word[2]);

Calendar cal=**new** GregorianCalendar(word0,(word1-1),word2);

**int** dayofyear=cal.get(Calendar.***DAY\_OF\_YEAR***);

//int num=dayofyear-31;

System.***out***.println("Day of year : "+dayofyear);

}

}

4. **Name Of the Day**

Given a date in the date format, write a program to get the day of the corresponding date .   
Include a class **UserMainCode** with a static method **displayDay** which accepts a date. In this method display the day of  given date . The return type is void.   
  
Create a Class **Main** which would be used to accept a date and call the static method present in UserMainCode.   
  
**Input and Output Format:**   
Input consists of a date.   
Refer sample output for formatting specifications.   
  
**Sample Input 1:**   
2011-10-21   
**Sample Output 1:**   
Friday   
  
**Sample Input 2:**   
2011-07-11   
**Sample Output 2:**   
Monday

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) **throws** Exception {

Scanner sc=**new** Scanner(System.***in***);

String str=sc.nextLine();

UserMainCode umc=**new** UserMainCode();

umc.displayDay(str);

}

}

**import** java.text.\*;

**import** java.util.\*;

**public** **class** UserMainCode {

**void** displayDay(String str) **throws** Exception{

SimpleDateFormat sdf=**new** SimpleDateFormat("yyyy-MM-dd");

Date date1=sdf.parse(str);

SimpleDateFormat sdf1=**new** SimpleDateFormat("EEEE");

str=sdf1.format(date1);

System.***out***.println(str);

}

}

5. **Before and After year**

Given a date , Write a program to get the date , before one year and after one year.

Include a class UserMainCode with a static method **displayDateDetails** which accepts a string. In this method display date in the format as given in sample input and output . The return type is void.  
  
 Create a Class Main which would be used to accept a string  and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of a string.

Refer sample output for formatting specifications.

**Sample Input :**

2016-10-23

**Sample Output :**

2017-10-23

2015-10-23

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) **throws** Exception {

Scanner sc=**new** Scanner(System.***in***);

String str=sc.nextLine();

UserMainCode umc=**new** UserMainCode();

umc.displayDay(str);

}

}

//import java.text.SimpleDateFormat;

**import** java.util.\*;

**import** java.text.\*;

**public** **class** UserMainCode {

**void** displayDay(String str)**throws** Exception{

SimpleDateFormat sdf=**new** SimpleDateFormat("yyyy-MM-dd");

Date date1=sdf.parse(str);

str=sdf.format(date1);

String word[]=str.split("-");

**int** word0=Integer.*parseInt*(word[0]);

**int** word1=Integer.*parseInt*(word[1]);

**int** word2=Integer.*parseInt*(word[2]);

Calendar cal=**new** GregorianCalendar((word0+1), (word1-1), word2);

Date date2=cal.getTime();

str=sdf.format(date2);

System.***out***.println(str);

Calendar cal1=**new** GregorianCalendar((word0-1), (word1-1), word2);

Date date3=cal1.getTime();

str=sdf.format(date3);

System.***out***.println(str);

}

}

6. **Day Of Month**

Given a year and month . Write a program to get the first day and last day of the month.   
Include a class UserMainCode with a static method **displayDay** which accepts two integer values.i.e year and month. In this method display the first day and last day of the given year and month. The return type is void.   
  
Create a Class Main which would be used to accept two integer values and call the static method present in UserMainCode.   
  
**Input and Output Format:**   
Input consists of a two integers.   
Refer sample output for formatting specifications.   
**Sample Input 1:**   
2015   
3   
**Sample Output 1:**   
SUNDAY   
TUESDAY   
  
**Sample Input 2:**   
1998   
7   
**Sample Output 2:**   
WEDNESDAY   
FRIDAY

**import** java.util.\*;

**import** java.text.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args)**throws** Exception {

Scanner sc=**new** Scanner(System.***in***);

**int** year=sc.nextInt();

**int** month=sc.nextInt();

UserMainCode ucm=**new** UserMainCode();

ucm.displayDay(year, month);

}

}

**import** java.util.\*;

**import** java.text.\*;

**public** **class** UserMainCode {

**void** displayDay(**int** year,**int** month) **throws** Exception

{

SimpleDateFormat sdfday=**new** SimpleDateFormat("EEEE");

//SimpleDateFormat sdfdate=new SimpleDateFormat("yyyy-MM-dd");

Date date=**new** Date();

Calendar c= Calendar.*getInstance*();

c.set(year,month-1,1);

String str1=sdfday.format(c.getTime());

System.***out***.println(str1.toUpperCase());

c.set(Calendar.***DAY\_OF\_MONTH***,c.getActualMaximum(Calendar.***DAY\_OF\_MONTH***));

String str2=sdfday.format(c.getTime());

System.***out***.println(str2.toUpperCase());

}

}

7. **Start and End day of the year**

Given a year as integer .Write a program to get the first day and last day of the of the given year

Include a class UserMainCode with a static method **displayDay** which accepts an integer. In this method display day in the format as given in sample input and output . The return type is void.  
  
 Create a Class Main which would be used to accept an integer and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of an integer.

Refer sample output for formatting specifications.  
**Sample Input And Output:**  
Enter the year  
**2015**  
Start Day of the given year is Thu  
End Day of the given year is Thu

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String[] args) **throws** Exception {

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("Enter the year");

**int** year =sc.nextInt();

UserMainCode umc=**new** UserMainCode();

umc.displayDay(year);

}

}

**import** java.text.SimpleDateFormat;

**import** java.util.Calendar;

**public** **class** UserMainCode {

**void** displayDay(**int** year) **throws** Exception

{

SimpleDateFormat sdf=**new** SimpleDateFormat("E");

Calendar c=Calendar.*getInstance*();

c.set(year,0,1);

String str1=sdf.format(c.getTime());

//System.out.println(c.getTime());

System.***out***.println("Start Day of the given year is "+str1);

c.set(year,11,31);

String str2=sdf.format(c.getTime());

//System.out.println(c.getTime());

System.***out***.println("End Day of the given year is "+str2);

}

}

8. **Difference between two years**

Given two dates, Write a program to get the difference between the 2 dates (in years and months)

Include a class UserMainCode with a static method **displayDateDetails** which accepts two strings. In this method. display the difference between the 2 dates in the format specified in the sample output . The return type is void.  
  
 Create a Class Main which would be used to accept two strings and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of two strings.

Refer sample output for formatting specifications.

**Sample Input :**

2015-05-15

2016-09-16

**Sample Output :**

**Difference between 2015-05-15 and 2016-09-16: 1 Years and 4 Months**

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String[] args) **throws** Exception {

Scanner sc=**new** Scanner(System.***in***);

String str1=sc.nextLine();

String str2=sc.nextLine();

UserMainCode umc=**new** UserMainCode();

umc.*displayDateDetails*(str1, str2);

}

}

**import** java.text.DateFormat;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**import** java.util.Calendar;

**public** **class** UserMainCode {

**static** **void** displayDateDetails(String n,String m)

{

**try**

{

SimpleDateFormat sdf=**new** SimpleDateFormat("yyyy-MM-dd");

Date date1=sdf.parse(n);

Date date2=sdf.parse(m);

Calendar c = Calendar.*getInstance*();

Calendar c1=Calendar.*getInstance*();

c.setTime(date1);

c1.setTime(date2);

**int** yeardiff=c1.get(Calendar.***YEAR***)-c.get(Calendar.***YEAR***);

**int** monthdiff=c1.get(Calendar.***MONTH***)-c.get(Calendar.***MONTH***);

System.***out***.println(" Difference between "+n+" and "+m+": "+yeardiff+" Years and "+monthdiff+" Months");

}

**catch** ( Exception ex ){

System.***out***.println(ex);

}

}

}

9. **Next and Previous Friday**

Write a program to get the next and previous Friday of the given date.

Include a class UserMainCode with a static method **DisplayDate** which accepts 3 integers. In this method display the 2 dates in the format as given in sample input and output . The return type is void.  
  
Create a Class Program which would be used to accept 3 integers that correspond to date and call the static method present in UserMainCode.  
  
**Input and Output Format:**  
Input consists of a string.  
Refer sample output for formatting specifications.  
All text in bold corresponds to input and the rest corresponds to output.

**Sample Input and Output:**

Enter year in Integer  
**2016**  
Enter Month in Integer  
**10**  
Enter date in Integer  
**23**  
Next Friday: 2016-10-28  
Previous Friday: 2016-10-21

**import** java.text.ParseException;

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String[] args) **throws** ParseException {

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("Enter year in Integer");

**int** year=sc.nextInt();

System.***out***.println("Enter Month in Integer");

**int** month=sc.nextInt();

System.***out***.println("Enter date in Integer");

**int** date=sc.nextInt();

UserMainCode umc=**new** UserMainCode();

umc.*DisplayDate*(year, month, date);

}

}

**import** java.lang.\*;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Calendar;

**import** java.util.Date;

**public** **class** UserMainCode {

**public** **static** **void** DisplayDate(**int** y,**int** m,**int** x)**throws** ParseException

{ **int** i;

Calendar cal=Calendar.*getInstance*();

cal.set(y,m-1,x);

**for**( i=0;i<7;i++)

{ cal.add(Calendar.***DATE***, +1);

Date pd=cal.getTime();

SimpleDateFormat sdf1 = **new** SimpleDateFormat("EEEE");

String result=sdf1.format(pd);

**if**(result.equals("Friday"))

{

SimpleDateFormat sdf2 = **new** SimpleDateFormat("YYYY-MM-dd");

String result1=sdf2.format(pd);

System.***out***.println("Next Friday: "+result1);

**break**;

}

}

cal.set(y,m-1,x);

**for**( i=0;i<7;i++)

{ cal.add(Calendar.***DATE***, -1);

Date pd1=cal.getTime();

SimpleDateFormat sdf4 = **new** SimpleDateFormat("EEEE");

String result3=sdf4.format(pd1);

**if**(result3.equals("Friday"))

{

SimpleDateFormat sdf3 = **new** SimpleDateFormat("YYYY-MM-dd");

String result2=sdf3.format(pd1);

System.***out***.println("Previous Friday: "+result2);

**break**;

}

}

}

}

10.

|  |
| --- |
| **Calculate Age**  Given two strings in the date format, write a program to calculate the age from the two dates.   Include a class UserMainCode with a static method displayAge which accepts two strings. In this method display age in the format as given in sample input and output . The return type is void.    Create a Class Main which would be used to accept two strings  and call the static method present in UserMainCode.   **Input and Output Format:**   Input consists of two  strings in specified format.  Refer sample output for formatting specifications.   **Sample Input :**  1994-03-24  2016-10-23   **Sample Output :**  I am 22 years, 6 months and 29 days old. |

Top of Form

Bottom of Form

**Days After and Before**

Given a date in the date format.Write a program to get date and day ten days before the given date and ten days after the given date.   
Include a class UserMainCode with a static method **displayDay** which accepts a string. In this method display the day and date in the format YYYY-MM-DD. The return type is void.   
  
Create a Class Main which would be used to accept a string and call the static method present in UserMainCode.   
  
**Input and Output Format:**   
Input consists of a string.   
Refer sample output for formatting specifications.   
  
**Sample Input 1:**   
2014-10-30   
  
**Sample Output 1:**   
2014-10-20   
Monday   
2014-11-09   
Sunday