ASSIGNMENT 1.3:

--------------------------------------------------------------------------------------------------------------------------------------

**1)**

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

**import** java.lang.Math;

**public** **class** ArmstrongNumber {

**public** **static** **void** armstrong(**int** n) {

**int** count=0,sum=0,temp,r;

temp=n;

**while**(temp!=0) {

temp=temp/10;

count=count+1;

}

temp=n;

**while**(temp!=0) {

r=temp%10;

**int** x=(**int**) Math.*pow*(r,count);

sum=sum+x;

temp=temp/10;

}

**if**(n==sum) {

System.***out***.println("It is an armstrong number");

}

**else** {

System.***out***.println("It is not an armstrong number");

}

}

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

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

System.***out***.println("enter the number:");

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

*armstrong*(n);

}

}

OUTPUT::

enter the number:

136

It is not an armstrong number

enter the number:

153

It is an armstrong number

2)

**import** java.util.Scanner;

**public** **class** Armclass {

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

**int** a=100;

**int** b=999;

**for**(**int** i=a;i<=b;i++) {

**int** sum=0,temp,r;

temp=i;

**while**(temp!=0) {

r=temp%10;

sum=sum+(r\*r\*r);

temp=temp/10;

}

**if**(sum==i) {

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

}

}

}

}

Output:

153

370

371

407

3)

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

**public** **class** Interest {

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

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

System.***out***.println("enter the principal:");

**float** p=sc.nextFloat();

System.***out***.println("enter the time:");

**float** t=sc.nextFloat();

System.***out***.println("enter the number of times:");

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

System.***out***.println("enter the rate:");

**float** r=sc.nextFloat();

**float** simpleInterest=((p\*t\*r)/100);

**double** compoundInterest=p\*(Math.*pow*((1+r/100),(t\*n)))-p;

System.***out***.println("simple Interest is:" +simpleInterest);

System.***out***.println("compound Interest is:" +compoundInterest);

}

}

Output:

enter the principal:

1000

enter the time:

3

enter the number of times:

1

enter the rate:

10

simple Interest is:300.0

compound Interest is:331.00008654594626

4)

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

**public** **class** SubjectMarks {

@SuppressWarnings("resource")

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

**int** x,y,z;

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

System.***out***.println("enter marks of x: ");

x=sc.nextInt();

System.***out***.println("enter marks of y: ");

y=sc.nextInt();

System.***out***.println("enter marks of z: ");

z=sc.nextInt();

**if**(x>60 && y>60 && z>60)

{

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

}

**else** **if**(x>60 && y>60 || y>60 && z>60 || x>60 && z>60)

{

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

}

**else** **if**(x>60 || y>60 || z>60 || x<60 && y<60 && z<60)

{

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

}

}

}

OUTPUT::

enter+ marks of x:

60

enter marks of y:

77

enter marks of z:

88

Promoted

5)

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

**public** **class** Taxpayment {

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

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

System.***out***.println("enter the gross annual salary:");

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

**double** Tax\_Amount;

**if**(salary<=180000) {

System.***out***.println("Tax amount: Nil ");

}

**else** **if**((salary>=181001)&&(salary<=300000)) {

Tax\_Amount=0.1\*salary;

System.***out***.println("Tax amount: " +Tax\_Amount);

}

**else** **if**((salary>=300001)&&(salary<=500000)) {

Tax\_Amount=0.2\*salary;

System.***out***.println("Tax amount: " +Tax\_Amount);

}

**else** **if**((salary>=500001)&&(salary<=1000000)) {

Tax\_Amount=0.3\*salary;

System.***out***.println("Tax amount: " +Tax\_Amount);

}

**else** {

System.***out***.println("no tax");

}

}

}

Output::

enter the gross annual salary:

500000

Tax amount: 100000.0

6)

**import** java.util.Scanner;

**public** **class** LoginPassword {

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

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

String sp = " ";

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

String username = sc.nextLine();

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

String password = sc.nextLine();

System.***out***.println(username + " you are Registered Successfully");

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

String name = sc.nextLine();

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

String passwordOne = sc.nextLine();

**if** (username.equals(name) && password.equals( passwordOne)) {

System.***out***.println("Welcome " + name + " you have Logged-in Successfully");

} **else** {

System.***out***.println("Username or password Mismatch");

}

**int** totalAttempts = 3;

**while** (totalAttempts != 0) {

**if** (name == "temp" && passwordOne == "pass") {

System.***out***.println("Login Correct!");

**return**;

} **else** {

System.***out***.println("Incorrect Login");

totalAttempts--;

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

}

}

**if** (totalAttempts == 0) {

System.***out***.println("Maximum number of attempts exceeded,,CONTACT ADMIN");

}

}

}

OUTPUT::

Enter the login name

ram

Enter the Password

123

ram you are Registered Successfully

Enter the Username

raj

Enter the Password

123

Username or password Mismatch

Incorrect Login

2

Incorrect Login

1

Incorrect Login

0

Maximum number of attempts exceeded,,CONTACT ADMIN

7)

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

**public** **class** Arrays {

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

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

System.***out***.println("enter the length:");

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

System.***out***.println("enter the element to search");

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

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

System.***out***.println("enter the elements:");

**int** flag=0;

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

a[i]=sc.nextInt();

}

System.***out***.println("print all the elements:");

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

**if**(ele==a[i]) {

System.***out***.println("element found in array");

flag=1;

**break**;

}

}

**if**(flag==0){

System.***out***.println("element not found in array");

}

}

}

OUTPUT::

enter the length:

15

enter the element to search

86

enter the elements:

5

12

14

6

78

19

1

23

26

35

37

7

52

86

47

print all the elements:

element found in array

8)

**public** **class** bubblesort {

**static** **void** bubbleSort(**int**[] a) {

**int** n = a.length;

**int** temp = 0;

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

**for**(**int** j=1; j < (n-i); j++){

**if**(a[j-1] > a[j]){

temp = a[j-1];

a[j-1] = a[j];

a[j] = temp;

}

}

}

}

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

**int** a[] ={5,12,14,6,78,19,1,23,26,35,37,7,52,86,47};

System.***out***.println("Array Before Bubble Sort");

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

System.***out***.print(a[i] + " ");

}

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

*bubbleSort*(a);

System.***out***.println("Array After Bubble Sort");

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

System.***out***.print(a[i] + " ");

}

}

}

OUTPUT::

Array Before Bubble Sort

5 12 14 6 78 19 1 23 26 35 37 7 52 86 47

Array After Bubble Sort

1 5 6 7 12 14 19 23 26 35 37 47 52 78 86

9)

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

**public** **class** TotalAvgMarks {

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

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

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

**int** m1[]=**new** **int**[3];

**int** m2[]=**new** **int**[3];

**int** m3[]=**new** **int**[3];

**int** totalsubjects[]=**new** **int**[3];

**int** totalstudents[]=**new** **int**[3];

**double** avgstudents[]=**new** **double**[3];

**double** avgSubjects[]=**new** **double**[3];

**int** i=0;

System.***out***.println("Enter the marks of 1st student");

**for**(i=0;i<3;i++) {

m1[i]=in.nextInt();

totalstudents[0]+=m1[i];

}

System.***out***.println("Enter the marks of 2nd student");

**for**(i=0;i<3;i++) {

m2[i]=in.nextInt();

totalstudents[1]+=m2[i];

}

System.***out***.println("Enter the marks of 3rd student");

**for**(i=0;i<3;i++) {

m3[i]=in.nextInt();

totalstudents[2]+=m3[i];

}

**for**(i=0;i<3;i++) {

totalsubjects[i]=m1[i]+m2[i]+m3[i];

}

**for**(i=0;i<3;i++) {

avgSubjects[i]=totalsubjects[i]/3;

}

**for**(i=0;i<3;i++) {

avgstudents[i]=totalstudents[i]/3;

}

System.***out***.println("Total and Average marks of 1st Student are "+totalstudents[0]+" and "+avgstudents[0]);

System.***out***.println("Total and Average marks of 2nd Student are "+totalstudents[1]+" and "+avgstudents[1]);

System.***out***.println("Total and Average marks of 3rd Student are "+totalstudents[2]+" and "+avgstudents[2]);

System.***out***.println("Total and Average marks of 1st Subject are "+totalsubjects[0]+" and "+avgSubjects[0]);

System.***out***.println("Total and Average marks of 2nd Subject are "+totalsubjects[1]+" and "+avgSubjects[1]);

System.***out***.println("Total and Average marks of 3rd Subject are "+totalsubjects[2]+" and "+avgSubjects[2]);

in.close();

}

}

OUTPUT::

Enter the marks of 1st student

90

75

50

Enter the marks of 2nd student

78

98

56

Enter the marks of 3rd student

78

65

90

Total and Average marks of 1st Student are 215 and 71.0

Total and Average marks of 2nd Student are 232 and 77.0

Total and Average marks of 3rd Student are 233 and 77.0

Total and Average marks of 1st Subject are 246 and 82.0

Total and Average marks of 2nd Subject are 238 and 79.0

Total and Average marks of 3rd Subject are 196 and 65.0

ASSIGNMENT 1.6::

1)

**package** prathibha;

**class** sample{

**private** **static** sample *obj*;

**private** sample(){ }

**public** **static** sample getInstance()

{

**if**(*obj*==**null**) {

*obj*=**new** sample();

}

**return** *obj*;

}

**public** **static** **void** getConnection()

{

System.***out***.println("singleton class");

}

}

**class** Singleton {

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

{

sample obj;

obj=sample.*getInstance*();

sample.*getConnection*();

}

}

Output::

Singleton class

2)

**package** javaassignment;

**class** employee

{

**int** incentive,overtime;

**public** **int** totalsal()

{

**return** 0;

}

}

**class** manager **extends** employee

{

**public** manager(**int** incentive) {

**this**.incentive=incentive;

}

**public** **int** totalsal()

{

**return** incentive;

}

}

**class** labour **extends** employee

{

**public** labour(**int** overtime) {

**this**.overtime=overtime;

}

**public** **int** totalsal()

{

**return** overtime;

}

}

**public** **class** methodoverriding {

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

{

manager m= **new** manager(200000);

labour l= **new** labour(300000);

System.***out***.println("total salary of employees: " + (m.totalsal()+l.totalsal()));

}

}

Output:

total salary of employees: 500000

3)

**class** bank {

**int** amount;

bank()

{

amount=0;

}

bank(**int** amount)

{

**this**.amount=amount;

}

**int** getAmount()

{

**return** amount;

}

}

**class** SavingsAccount **extends** bank {

SavingsAccount()

{

amount=0;

}

SavingsAccount(**int** deposit)

{

**super**(deposit);

}

**int** getAmount()

{

**return** (**super**.getAmount());

}

}

**class** CurrentAccount **extends** bank {

CurrentAccount()

{

**super**();

}

CurrentAccount(**int** credit)

{

**super**(credit);

}

**int** getSalary()

{

**return** (**super**.getAmount());

}

}

**public** **class** BankAccount {

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

{

SavingsAccount s= **new** SavingsAccount(200000);

System.***out***.println("Amount: " +s.getAmount());

CurrentAccount c= **new** CurrentAccount(300000);

System.***out***.println("Amount: " + c.getSalary());

**int** sum=s.getAmount()+c.getSalary();

System.***out***.println("Total Amount in the bank: " +sum);

}

}

Output:

Amount: 200000

Amount: 300000

Total Amount in the bank: 500000

4)

/\*\*

\* abstract method cannot have body

\* should be defined in a abstract class

\* **@author** HP

\*

\*/

**abstract** **class** AbstractClass {

**abstract** **public** **void** rollnumber();

}

/\*\*

\* abstract methods should be overridden to be used in childclass

\* or class should be declared as abstract

\* **@author** HP

\*

\*/

**class** student1 **extends** AbstractClass{

**public** **void** rollnumber() {

System.***out***.println(12);

}

}

**public** **class** Practice {

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

/\*\*

\* abstract class cannot be instantiated

\* **@param** args

\*/

student1 student=**new** student1();

student.rollnumber();

}

}

Output::

12

5)

/\*\*

\* abstract method cannot have body

\* should be defined in a abstract class

\* **@author** HP

\*

\*/

**abstract** **class** Shape {

**abstract** **public** **void** draw();

}

/\*\*

\* abstract methods should be overridden to be used in childclass

\* **@author** HP

\*

\*/

**class** Line **extends** Shape{

**public** **void** draw() {

System.***out***.println("It is a line");

}

}

**class** Rectangle **extends** Shape{

**public** **void** draw() {

System.***out***.println("It is a rectangle");

}

}

**class** Cube **extends** Shape{

**public** **void** draw() {

System.***out***.println("it is a cube");

}

}

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

/\*\*

\* abstract class cannot be instantiated

\* **@param** args

\*/

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

Line line=**new** Line();

Rectangle rectangle=**new** Rectangle();

Cube cube=**new** Cube();

line.draw();

rectangle.draw();

cube.draw();

}

}

OUTPUT::::::

It is a line

It is a rectangle

it is a cube

6)

**package** javaassignment;

**import** java.lang.reflect.Method;

**public** **abstract** **class** Persistence {

**abstract** **public** **void** persist();

}

**class** FilePersistence **extends** Persistence{

**public** **void** persist() {

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

}

}

**class** DatabasePersistence **extends** Persistence{

**public** **void** persist() {

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

}

}

**package** javaassignment;

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

**public** **class** ClientName {

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

**try** {

Class<?> p=Class.*forName*("javaassignment.Persistence");

Method m=p.getMethod("persist");

m.invoke(p);

}

**catch**(Exception e) { }

}

}

(Or Compile time we have::)

**abstract** **class** Persistence

{

**abstract** **void** persist();

}

**class** FilePersistence **extends** Persistence

{

**public** **void** persist() {

System.***out***.println("Data is being saved in File");

}

}

**class** DatabasePersistence **extends** Persistence

{

**public** **void** persist() {

System.***out***.println("Data is being saved in Database");

}

}

**public** **class** main {

**public** **static** **void** client(Persistence p) {

p.persist();

}

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

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

Persistence p1=**new** FilePersistence();

Persistence p2=**new** DatabasePersistence();

*client*(p1);

*client*(p2);

}

}

Output:

Data is being saved in File

Data is being saved in Database

7)

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

**public** **class** MainClass {

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

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

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

DessertItem candy=**new** Candy();

DessertItem cookie=**new** Cookie();

DessertItem icecream=**new** Icecream();

List<Integer> candies=**new** ArrayList<Integer>();

List<Integer> cookies=**new** ArrayList<Integer>();

List<Integer> icecreams=**new** ArrayList<Integer>();

**int** n,i,size;

**while**(**true**) {

System.***out***.println("Type o for Owner, c for customer and e for exit");

String type=in.next();

**if**(type.equals("o")) {

System.***out***.println("Enter the no.of candies to add into storage");

n=in.nextInt();

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

candies.add(1);

}

System.***out***.println("Enter the no.of cookies to add into storage");

n=in.nextInt();

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

cookies.add(1);

}

System.***out***.println("Enter the no.of icecreams to add into storage");

n=in.nextInt();

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

icecreams.add(1);

}

}

**else** **if**(type.equals("c")) {

System.***out***.println("Place the order:");

System.***out***.println("Enter the no.of candies");

n=in.nextInt();

size=candies.size()-1;

System.***out***.println(n+" "+size);

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

candies.remove(size-i);

}

**if**(n>0)

System.***out***.println("Cost for candies is "+(n\*candy.getCost())+" dollars");

System.***out***.println("Enter the no.of cookies");

n=in.nextInt();

size=cookies.size()-1;

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

cookies.remove(size-i);

}

**if**(n>0)

System.***out***.println("Cost for cookies is "+(n\*cookie.getCost())+" euros");

System.***out***.println("Enter the no.of icecreams");

n=in.nextInt();

size=icecreams.size()-1;

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

icecreams.remove(size-i);

}

**if**(n>0)

System.***out***.println("Cost for icecreams is "+(n\*icecream.getCost())+" rupees");

}

**else** {

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

**break**;

}

}

in.close();

}

}

Output:

Type o **for** Owner, c **for** customer and e **for** exit

o

Enter the no.of candies to add into storage

7

Enter the no.of cookies to add into storage

6

Enter the no.of icecreams to add into storage

7

Type o **for** Owner, c **for** customer and e **for** exit

o

Enter the no.of candies to add into storage

3

Enter the no.of cookies to add into storage

2

Enter the no.of icecreams to add into storage

8

Type o **for** Owner, c **for** customer and e **for** exit

c

Place the order:

Enter the no.of candies

2

2 9

Cost **for** candies is 4 dollars

Enter the no.of cookies

4

Cost **for** cookies is 4 euros

Enter the no.of icecreams

2

Cost **for** icecreams is 50 rupees

Type o **for** Owner, c **for** customer and e **for** exit

4

Exited

ASSIGNMENT 1.8::

1)

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

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

String str="Hello World";

**int** p=str.length();

System.***out***.println("The length of the string is: " +p);

}

}

OUTPUT::

The length of the string is: 11

2)

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

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

String str="Hello,";

String str2="How are you?";

String p=str.concat(str2);

System.***out***.println("The concatinated string is: " +p);

}

}

OUTPUT::

The concatinated string is: Hello,How are you?

3)

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

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

String str="Java String pool refers to collection of strings which are stored in heap memory";

System.***out***.println("string in lowercase:" +str.toLowerCase());

System.***out***.println("string in uppercase:" +str.toUpperCase());

System.***out***.println("string replacement:" +str.replace('a','$'));

System.***out***.println("collection is present?:" +str.contains("collection"));

String str1="java string pool refers to collection of strings which are stored in heap memory";

System.***out***.println("strings equal or not:" +str.equals(str1));

System.***out***.println("strings are equal:"+str.equalsIgnoreCase(str1));

}

}

OUTPUT::

string in lowercase:java string pool refers to collection of strings which are stored in heap memory

string in uppercase:JAVA STRING POOL REFERS TO COLLECTION OF STRINGS WHICH ARE STORED IN HEAP MEMORY

string replacement:J$v$ String pool refers to collection of strings which $re stored in he$p memory

collection is present?:true

strings equal or not:false

strings are equal:true

stringbuffer class::

1)

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

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

StringBuffer str = **new** StringBuffer("StringBuffer");

str.append(" is a peer class of string");

str.append(" that provides much of");

System.***out***.println("appended string :" +str.append("the functionality of strings."));

}

}

OUTPUT::

appended string :StringBuffer is a peer class of string that provides much ofthe functionality of strings.

2)

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

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

StringBuffer str = **new** StringBuffer("it is used to ");

System.***out***.println("new string :"+str.insert(14,"insert text"));

}

}

Output::

new string :it is used to insert text

3)

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

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

StringBuffer str = **new** StringBuffer("This method returns the reversed object on which it was called");

System.***out***.println("new string :"+str.reverse());

}

}

OUTPUT::

new string :dellac saw ti hcihw no tcejbo desrever eht snruter dohtem sihT

STRING BUILDER::::

1)

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

StringBuilder str = **new** StringBuilder("StringBuffer");

str.append(" is a peer class of string");

str.append(" that provides much of");

System.***out***.println("appended string :" +str.append("the functionality of strings."));

}

}

OUTPUT::

appended string :StringBuffer is a peer class of string that provides much ofthe functionality of strings.

2)

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

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

StringBuilder str = **new** StringBuilder("it is used to ");

System.***out***.println("new string :"+str.insert(14,"insert text"));

}

}

OUTPUT::

new string :it is used to insert text

3)

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

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

StringBuilder str = **new** StringBuilder("This method returns the reversed object on which it was called");

System.***out***.println("new string :"+str.reverse());

}

}

OUTPUT:

new string :dellac saw ti hcihw no tcejbo desrever eht snruter dohtem sihT

ASSIGNMENT 1.10::

1)

**public** **class** ExceptionOne {

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

**try** {

**int** a=10;

**int** b=0;

System.***out***.println(a/b);

} **catch** (Exception e) {

}

System.***out***.println("Exception is handled");

}

}

OUTPUT::

Exception is handled

3 a)

**public** **class** InsufficientBalanceException **extends** Exception {

**public** InsufficientBalanceException() {

}

}

**public** **class** SavingAccount{

**private** **long** id;

**private** **double** balance=0;

**void** Deposited(**double** x) {

balance+=x;

}

**void** Withdraw(**double** y) **throws** InsufficientBalanceException{

**if**(balance<y) {

**throw** **new** InsufficientBalanceException();

}

**else** {

balance-=y;

System.***out***.println("print the balance");

}

}

}

**package** exception.programs;

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

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

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

SavingAccount sa =**new** SavingAccount();

sa.Deposited(100);

**try** {

sa.Withdraw(1000);

} **catch** (InsufficientBalanceException e) {

e.printStackTrace();

}

}

}

OUTPUT::

exception.programs.InsufficientBalanceException

at exception.programs.SavingAccount.Withdraw(SavingAccount.java:11)

at exception.programs.Main.main(Main.java:10)

3 B)

**public** **class** InsufficientBalanceException **extends** Exception {

**public** InsufficientBalanceException() {

}

}

**public** **class** SavingAccount{

**private** **long** id;

**private** **double** balance=0;

**void** Deposited(**double** x) {

balance+=x;

}

**void** Withdraw(**double** y) **throws** IllegalBankTransactionException{

**if**(y<0) {

**throw** **new** IllegalBankTransactionException();

}

**else** {

balance-=y;

}

}

}

**package** exception.programs;

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

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

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

SavingAccount sa =**new** SavingAccount();

sa.Deposited(100);

**try** {

sa.Withdraw(-1);

} **catch** (IllegalBankTransactionException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

OUTPUT::

exception.programs.IllegalBankTransactionException

at exception.programs.SavingAccount.Withdraw(SavingAccount.java:11)

at exception.programs.Main.main(Main.java:10)

ASSIGNMENT 1.12::

1)

**package** generic.programs;

**import** java.util.HashSet;

**import** java.util.Map;

**import** java.util.Set;

**public** **class** Employee{

**int** id;

String name;

**int** salary;

String department;

**public** Employee(**int** id, String name, **int** salary, String department) {

**this**.id=id;

**this**.name=name;

**this**.salary=salary;

**this**.department=department;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** **int** getSalary() {

**return** salary;

}

**public** **void** setSalary(**int** salary) {

**this**.salary = salary;

}

**public** String getDepartment() {

**return** department;

}

**public** **void** setDepartment(String department) {

**this**.department = department;

}

@Override

**public** String toString() {

**return** "Employee id=" + id + ", name=" + name + ", salary=" + salary + ", department=" + department + "";

}

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

Set<Employee> employee= **new** HashSet<>();

employee.add(**new** Employee(1,"ram",20000,"It" ));

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

}

}

OUTPUT:::

[Employee id=1, name=ram, salary=20000, department=It]

2)

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

**import** java.util.HashMap;

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

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

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

HashMap<Integer, Double> h=**new** HashMap<Integer, Double>();

h.put(1,1.1);

h.put(2,2.1);

h.put(3,3.1);

h.put(4,4.1);

h.put(5,5.1);

h.put(6,6.1);

h.put(7,7.1);

h.put(8,8.1);

h.put(9,9.1);

h.put(10,10.1);

System.out.println("random values: " +h);

}

}

random values: {1=1.1, 2=2.1, 3=3.1, 4=4.1, 5=5.1, 6=6.1, 7=7.1, 8=8.1, 9=9.1, 10=10.1}

3)

**package** generic.programs;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Collections;

**import** java.util.List;

**public** **class** GenericMethod{

**public** **static** **final**<T> **void** swap(T[] x,**int** i,**int** j) {

T y=x[i];

x[i]=x[j];

x[j]=y;

}

**public** **static** **final** <T> **void** swap (List<T> l, **int** i, **int** j) {

Collections.<T>*swap*(l, i, j);

}

**private** **static** **void** test() {

String [] x = {"hii","hello"};

*swap*(x, 0, 1);

System.***out***.println("x:"+Arrays.*toString*(x));

List<String> l = **new** ArrayList<>(Arrays.*asList*(x));

*swap*(l, 0, 1);

System.***out***.println("l:"+l);

}

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

*test*();

}

}

OUTPUT::

x:[hello, hii]

l:[hii, hello]

4)

**public** **class** Pair<K,V> {

**private** K key;

**private** V value;

**public** K getKey() {

**return** key;

}

**public** **void** setKey(K key) {

**this**.key = key;

}

**public** V getValue() {

**return** value;

}

**public** **void** setValue(V value) {

**this**.value = value;

}

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

{

Pair<String,String> p=**new** Pair<String, String>();

p.setKey("1");

p.setValue("Hii");

System.***out***.println("key is: "+p.getKey() + " Value is: " +p.getValue());

Pair<String,java.util.Date> p1=**new** Pair<String, java.util.Date>();

p1.setKey("Today is");

p1.setValue(**new** java.util.Date());

System.***out***.println("key is: "+p1.getKey()+ " Value is: " +p1.getValue());

}

}

Output:

key is: 1 Value is: Hii

key is: Today is Value is: Wed Aug 11 13:54:19 IST 2021

1.14::

1)

**package** collections.programs;

**import** java.util.Collections;

**import** java.util.Iterator;

**import** java.util.Map;

**import** java.util.Map.Entry;

**import** java.util.Set;

**import** java.util.TreeMap;

**public** **class** TreeMain {

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

Contact c1= **new** Contact(224455675,"raji","raji@gmail.com","female");

Contact c2= **new** Contact(345634534,"raj","raj@gmail.com","male");

Contact c3= **new** Contact(454576764,"ram","ram@gmail.com","male");

Map<Long,Contact> LongContact = **new** TreeMap<>(Collections.*reverseOrder*());

LongContact.put((**long**) (1000456712), c1);

LongContact.put((**long**) (1000123512), c2);

LongContact.put((**long**) (1000123634), c3);

Set<Entry<Long,Contact>> entrySet =LongContact.entrySet();

Iterator<Entry<Long, Contact>> iterator = entrySet.iterator();

**while**(iterator.hasNext()) {

Map.Entry<Long, Contact> entry = (Map.Entry<Long,Contact>) iterator.next();

//Employee employee =entry.getKey();

Contact contact = entry.getValue();

System.***out***.print(entry.getKey()+" ");

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

}

}

}

**package** collections.programs;

**public** **class** Contact {

**private** **long** PhoneNumber;

**private** String Name;

**private** String Email;

**private** String Gender;

**public** Contact(**long** phoneNumber, String name, String email, String gender) {

**super**();

PhoneNumber = phoneNumber;

Name = name;

Email = email;

Gender = gender;

}

**public** **double** getPhoneNumber() {

**return** PhoneNumber;

}

**public** **void** setPhoneNumber(**long** phoneNumber) {

PhoneNumber = phoneNumber;

}

**public** String getName() {

**return** Name;

}

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

Name = name;

}

**public** String getEmail() {

**return** Email;

}

**public** **void** setEmail(String email) {

Email = email;

}

**public** String getGender() {

**return** Gender;

}

**public** **void** setGender(String gender) {

Gender = gender;

}

@Override

**public** String toString() {

**return** "Contact [PhoneNumber=" + PhoneNumber + ", Name=" + Name + ", Email=" + Email + ", Gender=" + Gender

+ "]";

}

}

OUTPUT:::

1000456712 Contact [PhoneNumber=224455675, Name=raji, Email=raji@gmail.com, Gender=female]

1000123634 Contact [PhoneNumber=454576764, Name=ram, Email=ram@gmail.com, Gender=male]

1000123512 Contact [PhoneNumber=345634534, Name=raj, Email=raj@gmail.com, Gender=male]

2)

**package** collections.programs;

**import** java.util.HashSet;

**import** java.util.Set;

**public** **class** Product {

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

Set<Integer> productId =**new** HashSet<>();

productId.add(1000);

productId.add(1001);

productId.add(1002);

productId.add(1003);

productId.add(1004);

productId.add(1005);

productId.add(1006);

productId.add(1007);

productId.add(1008);

productId.add(1009);

**for**(**int** product: productId) {

System.***out***.println(product + " ");

}

}

}

OUTPUT::

1008

1009

1000

1001

1002

1003

1004

1005

1006

1007

3)

4)

**package** collections.programs;

**import** java.time.LocalDate;

**import** java.util.LinkedList;

**import** java.util.List;

**public** **class** DateProgram {

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

LocalDate d1=LocalDate.*of*(2000, 12, 023);

//LocalDate d2=LocalDate.of(2001, 12, 023);

LinkedList<Object> list = **new** LinkedList<>();

list.add(d1);

**for**(Object i: list) {

**long** x,z;

**int** y=d1.getYear();

**if**(y!=0) {

x=(y%400==0)?(z=1):((y%100==0)?(z=0):((y%4==0)?(z=1):(z=0)));

**if**(x==1) {

System.***out***.println(d1+" is a leap year");

}

**else** {

System.***out***.println(d1+"is not a leap year");

}

}

}

}

}

OUTPUT::

2000-12-19 is a leap year

1.16 ASSIGNMENT

1)

**package** annotation.programs;

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Target;

@Target(ElementType.***METHOD***)

**@interface** Test

{

}

**class** Testcase{

@Test

**void** TestCase() {

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

}

}

**public** **class** AnnotationTest {

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

Testcase case1 =**new** Testcase();

case1.TestCase();

}

}

OUTPUT::

ANNOTATIONS

2)

**package** annotation.programs;

**import** java.lang.annotation.Annotation;

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

@Target(ElementType.***TYPE***)

@Retention(RetentionPolicy.***RUNTIME***)

**@interface** Info{

**int** Id();

String Date();

String Time();

**int** Version();

}

@Info(Date = "10-12-2021", Id = 1001, Time = "4:00", Version = 12)

**class** InfoTest{

String name;

String supervisor;

**public** InfoTest(String name, String supervisor) {

**super**();

**this**.name = name;

**this**.supervisor = supervisor;

}

**void** display() {

System.***out***.println("Annotations are printed");

}

}

**public** **class** MainTest {

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

InfoTest obj = **new** InfoTest("Athen", "Simon");

obj.display();

Class c=obj.getClass();

System.***out***.println("Dvelopers name: " +obj.name);

System.***out***.println("Developers Supervisor :" +obj.supervisor);

Annotation an=c.getAnnotation(Info.**class**);

Info s = (Info)an;

System.***out***.println("Developers ID: " +s.Id());

System.***out***.println("Date: " +s.Date());

System.***out***.println("Time :" +s.Time());

System.***out***.println("Version: " +s.Version());

}

}

OUTPUT::

Annotations are printed

Dvelopers name: Athen

Developers Supervisor :Simon

Developers ID: 1001

Date: 10-12-2021

Time :4:00

Version: 12

3)

package annotation.programs;

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

import java.lang.reflect.Method;

public class MyClass {

@Target(value=ElementType.METHOD)

@Retention(RetentionPolicy.RUNTIME)

@interface Execute{

int sequence();

}

@Execute(sequence = 2)

public void myMethod() {

System.out.println("sequence 2");

}

@Execute(sequence = 1)

public void myMethod2() {

System.out.println("sequence 1");

}

@Execute(sequence = 3)

public void myMethod3() {

System.out.println("sequence 3");

}

public static void main(String[] args) {

// TODO Auto-generated method stub

MyClass m=new MyClass();

Method[] methods = m.getClass().getMethods();

for(Method method : methods) {

Execute an = method.getAnnotation(Execute.class);

if(an!= null)

try {

method.invoke(m);

}catch(Exception e) {

e.printStackTrace();

}

}

}

OUTPUT::

sequence 3

sequence 1

sequence 2

1.18 ASSIGNMENT:::

1)

**package** lambda.programs;

**interface** Arithmetic{

**int** numbers(**int** a, **int** b);

}

**public** **class** ArithmaticOperations {

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

Arithmetic addition =(**int** a, **int** b) ->(a+b);

System.***out***.println("Addition of two numbers: " +addition.numbers(5, 5));

Arithmetic subtraction =(**int** a, **int** b) ->(a-b);

System.***out***.println("Subtraction of two numbers: " +subtraction.numbers(8, 4));

Arithmetic multiplication =(**int** a, **int** b) ->(a\*b);

System.***out***.println("Multiplication of two numbers: " +multiplication.numbers(6, 2));

Arithmetic division =(**int** a, **int** b) ->(a/b);

System.***out***.println("Division of two numbers: " +division.numbers(12, 3));

}

}

OUTPUT::

Addition of two numbers: 10

Subtraction of two numbers: 4

Multiplication of two numbers: 12

Division of two numbers: 4

2)

**package** lambda.programs;

**interface** Order{

**void** price();

}

**public** **class** OrdersProgram {

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

**int** a=1000000;

Order item = ()->{

**if**(a > 10000) {

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

}

};

item.price();

}

}

OUTPUT::

ACCEPTED

3)

**package** lambda.programs;

// PERSON CLASS

**public** **class** Person {

**private** String firstname;

**private** String lastname;

**private** **int** age;

**public** Person(String firstname, String lastname, **int** age) {

**super**();

**this**.firstname = firstname;

**this**.lastname = lastname;

**this**.age = age;

}

**public** String getFirstname() {

**return** firstname;

}

**public** **void** setFirstname(String firstname) {

**this**.firstname = firstname;

}

**public** String getLastname() {

**return** lastname;

}

**public** **void** setLastname(String lastname) {

**this**.lastname = lastname;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

@Override

**public** String toString() {

**return** "Person [firstname=" + firstname + ", lastname=" + lastname + ", age=" + age + "]";

}

}

//suplier class

**import** java.util.function.Supplier;

**public** **class** SuplierTest {

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

Supplier < Person > supplier =() -> { **return** **new** Person("Charles", "Dickens", 60);

};

Person p = supplier.get();

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

}

}

OUTPUT::

Person [firstname=Charles, lastname=Dickens, age=60]

**package** lambda.programs;

//PREDICATE AND CONSUMER

**import** java.util.Arrays;

**import** java.util.List;

**import** java.util.function.Consumer;

**import** java.util.function.Predicate;

**public** **class** PredicateAndConsumer {

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

List<Person> people =Arrays.*asList*(

**new** Person("Charles","dickens",60),

**new** Person("Char","dens",50),

**new** Person("arles","kens",80)

);

System.***out***.println("print all persons");

*Perform*(people, p->**true**, p->System.***out***.println(p));

}

**public** **static** **void** Perform(List<Person> people, Predicate<Person> predicate, Consumer<Person> consumer)

{

**for**(Person p: people)

{

consumer.accept(p);

}

}

}

OUTPUT::

print all persons

Person [firstname=Charles, lastname=dickens, age=60]

Person [firstname=Char, lastname=dens, age=50]

Person [firstname=arles, lastname=kens, age=80]

//FUNCTION

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**import** java.util.function.Function;

**public** **class** FunctionProgram {

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

Function<Person, String> function= (Person p)-> {**return** p.getFirstname();};

List<Person> people=Arrays.*asList*(

**new** Person("Charles", "Dickens", 45),

**new** Person("Lewis", "Caroll", 45),

**new** Person("Thomas", "Carlyle", 45),

**new** Person("Charolotte", "Bronte", 45)

);

System.***out***.println("print all persons");

List<Person> name=*listperson*(people, Function.*identity*());

name.forEach(System.***out***::println);

}

**public** **static** List<Person> listperson(List<Person> people, Function<Person, Person> m)

{

List<Person> name=**new** ArrayList<Person>();

**for**(Person p: people)

{

name.add(m.apply(p));

}

**return** name;

}

}

OUTPUT::

print all persons

Person [firstname=Charles, lastname=Dickens, age=45]

Person [firstname=Lewis, lastname=Caroll, age=45]

Person [firstname=Thomas, lastname=Carlyle, age=45]

Person [firstname=Charolotte, lastname=Bronte, age=45]

4)

**package** lambda.programs;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** OddOne {

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

List<String> list = **new** ArrayList<>();

list.add("prayag");

list.add("phalguni");

list.add("tom");

list.removeIf(m->(m.length()%2==1));

System.***out***.println("words with even length: ");

**for**(String s:list) {

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

}

}

}

OUTPUT::

words with even length:

prayag

phalguni

5)

**package** lambda.programs;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** FirstLetter {

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

List<String> list = **new** ArrayList<>();

list.add("Ram");

list.add("Emmie");

list.add("Polar");

list.add("Fun");

StringBuilder Letters = **new** StringBuilder();

list.forEach(n->Letters.append( n.substring(0,1)));

System.***out***.println("Print the letters: " +Letters.toString( ) );

}

}

OUTPUT::

Print the letters: REPF

6)

**package** lambda.programs;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.function.UnaryOperator;

**public** **class** ReplaceWordProgram {

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

ArrayList<String> words = **new** ArrayList<>(Arrays.*asList*("lambda","expressions"));

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

words.replaceAll( **new** RepalceAll() );

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

}

}

**class** RepalceAll **implements** UnaryOperator<String>

{

@Override

**public** String apply(String t) {

**return** t.toUpperCase();

}

}

OUTPUT::

[lambda, expressions]

[LAMBDA, EXPRESSIONS]

7)

**package** lambda.programs;

**import** java.util.Map;

**import** java.util.TreeMap;

**public** **class** AppendProgram {

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

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

map.put("Ram",1);

map.put("Emmie",2);

map.put("Polar",3);

map.put("Fun",5);

StringBuilder Result = **new** StringBuilder();

map.forEach((k,v)->Result.append(k).append(v));

System.***out***.println("Print the values appended: " +Result.substring(0,Result.length()).toString( ) );

}

}

OUTPUT::

Print the values appended: Emmie2Fun5Polar3Ram1

8)

**package** lambda.programs;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.function.Consumer;

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

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

{

Consumer<List<Integer> >

displayList = list -> list.stream().forEach(a -> System.***out***.print(a + " "));

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

list.add(1);

list.add(2);

list.add(3);

list.add(4);

list.add(5);

list.add(6);

displayList.accept(list);

}

}

OUTPUT::

1 2 3 4 5 6

1.19 ASSIGNMENT::

1)

**package** lambda.streams.programs;

**import** java.util.Arrays;

**import** java.util.Comparator;

**import** java.util.List;

**public** **class** MainClass {

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

List<Fruit> fruits = Arrays.*asList*(

**new** Fruit("mango", 250,"orange"),

**new** Fruit("apple",100,"red"),

**new** Fruit("banana",200 ,"yellow"),

**new** Fruit("guava",50 ,"green"),

**new** Fruit("cherries",10,"red")

);

fruits.stream()

.filter(f->f.getCalories()<100)

.sorted(Comparator.*comparingInt*(Fruit::getCalories).reversed())

.forEach(name->System.***out***.println(name));

}

}

OUTPUT::

Fruit [name=guava, calories=50, color=green]

Fruit [name=cherries, calories=10, color=red]

2)

**package** lambda.streams.programs;

**import** java.util.Arrays;

**import** java.util.Comparator;

**import** java.util.List;

**public** **class** MainClass {

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

List<Fruit> fruits = Arrays.*asList*(

**new** Fruit("mango", 250,"orange"),

**new** Fruit("apple",100,"red"),

**new** Fruit("banana",200 ,"yellow"),

**new** Fruit("guava",50 ,"green"),

**new** Fruit("cherries",10,"red")

);

fruits.stream()

.forEach(x->System.***out***.println(x.getColor()));

}

}

OUTPUT::

orange

red

yellow

green

red

3)

**package** lambda.streams.programs;

**import** java.util.Arrays;

**import** java.util.Comparator;

**import** java.util.List;

**public** **class** MainClass {

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

List<Fruit> fruits = Arrays.*asList*(

**new** Fruit("mango", 250,"orange",100),

**new** Fruit("apple",100,"red",200),

**new** Fruit("banana",200 ,"yellow",60),

**new** Fruit("guava",50 ,"green",80),

**new** Fruit("cherries",10,"red",150)

);

fruits.stream()

.filter(f->f.getColor().equals("red"))

.sorted(Comparator.*comparingInt*(Fruit::getPrice))

.forEach(name->System.***out***.println(name));

}

}

OUTPUT::

Fruit [name=cherries, calories=10, color=red, price=150]

Fruit [name=apple, calories=100, color=red, price=200]

4)

TRADER PROGRAMS::

package lambda.streams.programs;

import java.util.Arrays;

import java.util.Comparator;

import java.util.List;

public class MainClass {

public static void main(String[] args) {

List<Trader> list=new ArrayList<>();

list.add(new Trader("pooja", "pune"));

list.add(new Trader("ram", "delhi"));

list.add(new Trader("Dia", "pune"));

list.add(new Trader("raj", "delhi"));

list.add(new Trader("Charmi", "indore"));

list.stream()

.distinct()

.forEach(p->System.out.println(p.getCity()));

System.out.println("-------------------------------");

list.stream()

.filter(f->f.getCity().matches("pune"))

.forEach(c->System.out.println(c.getName()));

System.out.println("-------------------------------");

list.stream()

.sorted( Comparator.comparing(n->n.toString()))

.forEach(c->System.out.println(c.getName()));

System.out.println("-------------------------------");

list.stream()

.filter(f->f.getCity().matches("indore"))

.forEach(c->System.out.println(c.getName()));

}

}

OUTPUT::

pune

delhi

pune

delhi

indore

-------------------------------

pooja

Dia

-------------------------------

pooja

raj

Charles

Dia

ram

-------------------------------

Charmi

TRANSACTION PROGRAMS::

import java.util.ArrayList;

import java.util.Comparator;

import java.util.List;

import java.util.Optional;

public class TransactionMain {

private static Trader Trader;

public static void main(String[] args)

{

List<Transaction> list1=new ArrayList<>();

list1.add(new Transaction(Trader,2011,1));

list1.add(new Transaction(Trader,2016,2));

list1.add(new Transaction(Trader,2017,3));

list1.add(new Transaction(Trader,2018,4));

list1.add(new Transaction(Trader,2019,5));

list1.stream()

.filter(t->t.getYear()==2011)

.sorted(Comparator.comparing(Transaction::getValue))

.forEach(c->System.out.println(c.getValue()));

System.out.println("<--------------------------------------------->");

System.out.println("<--------------------------------------------->");

Optional<Transaction> tran= list1.stream()

.max(Comparator.comparing(Transaction::getValue));

System.out.println(tran);

System.out.println("<--------------------------------------------->");

Optional<Transaction> trans= list1.stream()

.min(Comparator.comparing(Transaction::getValue));

System.out.println(trans);

}

}

Output:

1

4

<--------------------------------------------->

Optional[Transaction [trader=null, year=2019, value=5]]

<--------------------------------------------->

Optional[Transaction [trader=null, year=2011, value=1]]