Core java topic vice programs

1.Arrays

**Array:** Array is the most important thing in any programming language. By definition, array is the static memory allocation. It allocates the memory for the same data type in sequence. It contains multiple values of same types. It also store the values in memory at the fixed size. Multiple types of arrays are used in any programming language such as:  
one – dimensional, two – dimensional or can say multi – dimensional.

**Advantage of Array**

Code Optimization: It makes the code optimized, we can retrieve or sort the data easily.  
Random access: We can get any data located at any index position.  
 **Disadvantage of Array**

Size Limit: We can store only fixed size of elements in the array. It doesn’t grow its size at runtime. To solve this problem, collection framework is used in java.

int rno;

int rno1;

int rno2;

.

.

.

int rno49

**Declaring Array Variables:**  
To use an array in a program, you must declare a variable to reference the array, and you must specify the type of array the variable can reference. Here is the syntax for declaring an array variable:

dataType[] arrayVar; // preferred way.

or

dataType arrayVar[]; // works but not preferred way.

Example: int[] myList; // preferred way.

or

int myList[]; // works but not preferred way.

**Creating Arrays:**

You can create an array by using the new operator with the following syntax:

arrayVar = new dataType[arraySize]; //one way

dataType[] arrayVar = new dataType[arraySize]; //second way

dataType[] arrayVar = {value0, value1, ..., valuek}; //3 way

Examples by using array :

Here is a complete example of showing how to create, initialize and process arrays:

public class TestArray {

public static void main(String[] args) {

double[] myList = {1.9, 2.9, 3.4, 3.5};

// Print all the array elements

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

System.out.println(myList[i] + " ");

}

// Summing all elements

double total = 0;

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

total += myList[i];

}

System.out.println("Total is " + total);

// Finding the largest element

double max = myList[0];

for (int i = 1; i < myList.length; i++) {

if (myList[i] > max) max = myList[i];

}

System.out.println("Max is " + max);

}

}

Output:

1.9

2.9

3.4

3.5

Total is 11.7

Max is 3.5  
---------------------------------------------------------------------------------------------------------

Example 2:

The following code displays all the elements in the array myList:

public class TestArray {

public static void main(String[] args) {

double[] myList = {1.9, 2.9, 3.4, 3.5};

// Print all the array elements

for (double element: myList) {

System.out.println(element);

}

}

}

Output :

1.9

2.9

3.4

3.5

**Multidimensional array:**

In such case, data is stored in row and column based index (also known as matrix form).  
**Syntax to Declare Multidimensional Array in java:**

dataType[][] arrayRefVar; (or)

dataType [][]arrayRefVar; (or)

dataType arrayRefVar[][]; (or)

dataType []arrayRefVar[];

**Example to initantiate Multidimensional Array in java**

int[][] arr=new int[3][3];//3 row and 3 column

**Example to initialize Multidimensional Array in java**

arr[0][0]=1;

arr[0][1]=2;

arr[0][2]=3;

arr[1][0]=4;

arr[1][1]=5;

arr[1][2]=6;

arr[2][0]=7;

arr[2][1]=8;

arr[2][2]=9;

Example for Multidimensional:

Let’s see the simple example to declare, instantiate, initialize and print the 2Dimensional array.

class B{

public static void main(String args[]){

//declaring and initializing 2D array

int arr[][]={{1,2,3},{2,4,5},{4,4,5}};

//printing 2D array

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

for(int j=0;j<3;j++){

System.out.print(arr[i][j]+" ");

}

System.out.println();

}

}

}

Output:

1 2 3

2 4 5

4 4 5

Topic 2.Inheritence:

Types of Inheritence

===================================================================

1.single level

2.multi level

3.hirarichal

4.multiple level

5.hybrid

=================================================================

1.single level

parent class=>child class

PARENT CALSS

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

class Employee{

float salary=40000;

}

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

CHILD CLASS

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

class Programmer extends Employee{

int bonus=10000;

public static void main(String args[]){

Programmer p=new Programmer();

System.out.println("Programmer salary is:"+p.salary);

System.out.println("Bonus of Programmer is:"+p.bonus);

}

}

=============================================

2.MULTILEVEL

============================================

parent class=>child class=> grand son

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

parent class

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

class Employee{

float salary=40000;

}

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

CHILD CLASS

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

class Programmer extends Employee{

int bonus=10000;

public static void main(String args[]){

Programmer p=new Programmer();

System.out.println("Programmer salary is:"+p.salary);

System.out.println("Bonus of Programmer is:"+p.bonus);

}

}

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

grand son class

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

class Programmer2 extends Employee{

int salary=20000;

public static void main(String args[]){

Programmer2 p2=new Programmer2();

System.out.println("Programmer salary is:"+p2.salary);

System.out.println("Bonus of Programmer is:"+p2.bonus);

}

}

==================================================

example 2 for multi level

package Multi;

public class Father {

int money=100000;

protected int wt=65;

public int getWt() {

return wt;

}

public void setWt(int wt) {

this.wt = wt;

}

private String gf="samantha"; // restricted

public String getGf() {

return gf;

}

public void setGf(String gf) {

this.gf = gf;

}

void isDrink() {

System.out.println("drinking is good habit");

}

public static void main(String[] args) {

}

}

=====================

child classs

=============

package Multi;

public class Son extends Father{

String name;

protected int ht=6;

public int getHt() {

return ht;

}

public void setHt(int ht) {

this.ht = ht;

}

public void longDrive() {

System.out.println(name+"i am going long drive with gf");

}

public static void main(String[] args) {

Son s1=new Son();

s1.money=20000;

System.out.println(s1.money);

s1.isDrink();

s1.name="sai";

s1.longDrive();

int w1=s1.getWt();

System.out.println("my weight is"+w1);

String s11=s1.getGf();

System.out.println("my gf is "+s11);

}

}

================================

granson class

===========================

package Multi;

public class GrandSon extends Son {

int age;

public void play() {

System.out.println("grandchild is playing game");

}

public static void main(String[] args) {

GrandSon g=new GrandSon();

g.name="ram";

System.out.println(g.name);

g.isDrink();

int ht1=g.getHt();

System.out.println("my height is "+ ht1);

}

}

====================================================================================

3.hirarical inheritence

parent class

==========================

class Animal{

void eat(){System.out.println("eating...");}

}

child class

===================

class Dog extends Animal{

void bark(){System.out.println("barking...");}

}

child class

=====================

class Cat extends Animal{

void meow(){System.out.println("meowing...");}

}

===================

testing program

====================

class TestInheritance3{

public static void main(String args[]){

Cat c=new Cat();

c.meow();

c.eat();

//c.bark();//C.T.Error

}}

=========================================================

4.multiple level

===========================================================

inter face1

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

public interface Student {

public void student\_name(String sname);

public void student\_age(int age);

}

================

inter face 2

================

public interface Trainer extends Student {

public void head();

public void trainer();

}

===================================================

child classs

===================================================

public class Sai implements Student,Trainer {

private static int age;

private static String sname;

public static void main(String[] args) {

Sai s=new Sai();

s.student\_age(age);

s.student\_name(sname);

s.head();

s.trainer();

}

@Override

public void student\_name(String sname) {

String sname1="sai";

System.out.println("my name is :"+sname1);

}

@Override

public void student\_age(int age) {

int age1=22;

System.out.println("my age is : "+ age1);

}

@Override

public void head() {

// TODO Auto-generated method stub

System.out.println("my hr is raja sir");

}

@Override

public void trainer() {

// TODO Auto-generated method stub

System.out.println("my hr is bhavna");

}

}

=========================================================================================

another example for multiple:

=========================================================================================

class A{

void msg(){System.out.println("Hello");}

}

class B{

void msg(){System.out.println("Welcome");}

}

class C extends A,B{

public static void main(String args[]){

C obj=new C();

obj.msg();

}

}

=========================================================================================

5.hybrid level

=============================================================

child class

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

public class ClassA

{

public void dispA()

{

System.out.println("disp() method of ClassA");

}

}

parent interface1

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

public interface InterfaceB

{

public void show();

}

===================

parent interface 2

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

public interface InterfaceC

{

public void show();

}

============================================================

grandchild it extends child classand implements interface 1 and interface2

=============================================================================

public class ClassD extends ClassA implements InterfaceB,InterfaceC

{

public void show()

{

System.out.println("show() method implementation");

}

public void dispD()

{

System.out.println("disp() method of ClassD");

}

public static void main(String args[])

{

ClassD d = new ClassD();

d.dispD();

d.show();

}

}

3 Polymorphism

The derivation of the word Polymorphism is from two different Greek words- poly and morphs. “Poly” means numerous, and “Morphs” means forms. So polymorphism means innumerable forms. Polymorphism is one of the most significant features of Object-Oriented Programming.

Examples:

Example 1: A superclass named “Shapes” has a method “area()”. Subclasses of “Shapes” can be “Triangle”, “circle”, “Rectangle”, etc. Each subclass has its way of calculating area. Using Inheritance and Polymorphism, the subclasses can use the “area()” method to find the area’s formula for that shape.

class Shapes {

  public void area() {

    System.out.println("The formula for area of ");

  }

}

class Triangle extends Shapes {

  public void area() {

    System.out.println("Triangle is ½ \* base \* height ");

  }

}

class Circle extends Shapes {

  public void area() {

    System.out.println("Circle is 3.14 \* radius \* radius ");

  }

}

class Main {

  public static void main(String[] args) {

    Shapes myShape = new Shapes();  // Create a Shapes object

    Shapes myTriangle = new Triangle();  // Create a Triangle object

    Shapes myCircle = new Circle();  // Create a Circle object

    myShape.area();

    myTriangle.area();

    myShape.area();

    myCircle.area();

  }

}

Output :

The formula for area of Triangle is ½ \* base \* height  
The formula for area of Circle is 3.14 \* radius \* radius

=====================================================================

**Topic 4 : abstract class**

**What is abstract class in Java?**

Use the **abstract** keyword to declare a class abstract. An abstract class is something which is incomplete and you can not create instance of abstract class. If you want to use it you need to make it complete or concrete by extending it. A class is called concrete if it does not contain any abstract method and implements all abstract method inherited from abstract class or interface it has implemented or extended. By the way Java has concept of abstract classes, abstract method but a variable can not be abstract in Java.

Example 1:

abstract class Shape{

abstract void draw();

}

class Rectangle extends Shape{

void draw(){

System.out.println("Drawing Rectangle");

}

}

class Traingle extends Shape{

void draw(){

System.out.println("Drawing Traingle");

}

}

class AbstractDemo{

public static void main(String args[]){

Shape s1 = new Rectangle();

s1.draw();

s1 = new Traingle();

s1.draw();

}

}

Output :

Drawing Rectangle

Drawing Traingle

**clsName**is a valid identifier in java. It is a class name.  
**abstract**is a keyword to define that method an abstract method.  
**rtype**is return type of a method.  
**mthName**is a method name and valid java identifier

**Declaring a method as abstract has two results:**

* The class must also be declared abstract. If a class contains an abstract method, the class must be abstract as well.
* Any child class must either override the abstract method or declare itself abstract.

Example 2:

**package** abstractclass;

**abstract** **class** Mobile {

**abstract** **void** myMobile();

**abstract** **void** price();

}

**package** abstractclass;

**public** **class** Iphone **extends** Mobile {

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

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

Iphone i=**new** Iphone();

i.myMobile();

i.price();

}

@Override

**void** myMobile() {

System.***out***.println(" my mobile is iphone");

}

@Override

**void** price() {

System.***out***.println("my mobile price is 200000");

}

}

Output:

my mobile is iphone

my mobile price is 200000

topic 5 : Interface

interface some points :

* The class which implements the interface needs to provide functionality for the methods declared in the interface
* All methods in an interface are implicitly public and abstract
* An interface cannot be instantiated
* An interface reference can point to objects of its implementing classes
* An interface can extend from one or many interfaces. A class can extend only one class but implement any number of interfaces

Example :

Program1:

**public** **interface** Student {

**public** **void** student\_name(String sname);

**public** **void** student\_age(**int** age);

}

Program 2:

**public** **interface** Trainer **extends** Student {

**public** **void** head();

**public** **void** trainer();

}

Program 3:

**public** **class** Sai **implements** Student,Trainer {

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

**private** **static** String *sname*;

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

Sai s=**new** Sai();

s.student\_age(*age*);

s.student\_name(*sname*);

s.head();

s.trainer();

}

@Override

**public** **void** student\_name(String sname) {

String sname1="sai";

System.***out***.println("my name is :"+sname1);

}

@Override

**public** **void** student\_age(**int** age) {

**int** age1=22;

System.***out***.println("my age is : "+ age1);

}

@Override

**public** **void** head() {

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

System.***out***.println("my hr is raja sir");

}

@Override

**public** **void** trainer() {

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

System.***out***.println("my hr is bhavna");

}

}

Output :

my age is : 22

my name is :sai

my hr is raja sir

my hr is bhavna

6 : Exception Handling

**package** Except;

**public** **class** Program {

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

**try** {

**int** a[]= {1,2,3};

a[4]=5;

**int** i=3,j=2;

**int** k=i/j;

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

}

**catch**(ArithmeticException e) {

System.***err***.println("arthematic exception "+e.getMessage());

}

**catch**(ArrayIndexOutOfBoundsException e) {

System.***err***.println("array index exception limit ");

}

**catch**(Exception e) {

System.***err***.println("error");

}**finally**{

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

}

}

}

7.Multithreading

Example:

Programs:1

**package** multith;

**class** MulthiTh **implements** Runnable {

**private** Thread t;

**private** String threadName;

// constructor

MulthiTh( String name) {

threadName = name;

System.***out***.println("Creating " + threadName );

}

// run method

**public** **void** run() {

System.***out***.println("Running " + threadName );

**try** {

**for**(**int** i = 4; i > 0; i--) {

System.***out***.println("Thread: " + threadName + ", " + i);

// Let the thread sleep for a while.

Thread.*sleep*(50);

}

} **catch** (InterruptedException e) {

System.***out***.println("Thread " + threadName + " interrupted.");

}

System.***out***.println("Thread " + threadName + " exiting.");

}

//start

**public** **void** start () {

System.***out***.println("Starting " + threadName );

**if** (t == **null**) {

t = **new** Thread (**this**, threadName);

t.start ();

}

}

}

Program 2:

**package** multith;

**public** **class** TestDemo {

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

MulthiTh R1 = **new** MulthiTh( "sai-1");

R1.start();

MulthiTh R2 = **new** MulthiTh( "sai-2");

R2.start();

Reverse r=**new** Reverse("sai");

r.start();

}

}

Example 2:

Program 1:

**package** multith;

**public** **class** School {

String name;

**void** List(String name) {

**this**.name=name;

}}

**class** MyThread11 **extends** Thread{

School name;

**public** MyThread11(School name) {

**this**.name=name;

}

**public** **void** run() {

System.***out***.println("bhavna trainer");

}

}

**class** MyThread22 **extends** Thread{

School name;

**public** MyThread22(School name) {

**this**.name=name;

}

**public** **void** run() {

System.***out***.println("sai student");

}

}

Program: 2

**package** multith;

**public** **class** Test {

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

School sch=**new** School();

MyThread11 m1=**new** MyThread11(sch);

MyThread22 m2=**new** MyThread22(sch);

m1.start();

m2.start();

}

}

Program 3:

**package** multith;

**public** **class** thread {

}

Synchronization in Java

Synchronization in java is the capability *to control the access of multiple threads to any shared resource*.

Java Synchronization is better option where we want to allow only one thread to access the shared resource.

The synchronization is mainly used to

1. To prevent thread interference.
2. To prevent consistency problem.

There are two types of synchronizationOOPs Concepts in Java

1. Process Synchronization
2. Thread Synchronization

### **Thread Synchronization**

There are two types of thread synchronization mutual exclusive and inter-thread communication.

1. Mutual Exclusive
   1. Synchronized method.
   2. Synchronized block.
   3. static synchronization.
2. Cooperation (Inter-thread communication in java)

### **Mutual Exclusive**

Mutual Exclusive helps keep threads from interfering with one another while sharing data. This can be done by three ways in java:

1. by synchronized method
2. by synchronized block
3. by static synchronization

example 1: In this example, there is no synchronization, so output is inconsistent

**class** Table{

**void** printTable(**int** n){//method not synchronized

**for**(**int** i=1;i<=5;i++){

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

**try**{

Thread.*sleep*(400);

}**catch**(Exception e){System.***out***.println(e);}

}

}

}

**class** MyThread1 **extends** Thread{

Table t;

MyThread1(Table t){

**this**.t=t;

}

**public** **void** run(){

t.printTable(5);

}

}

**class** MyThread2 **extends** Thread{

Table t;

MyThread2(Table t){

**this**.t=t;

}

**public** **void** run(){

t.printTable(100);

}

}

**class** TestSynchronization1{

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

Table obj = **new** Table();//only one object

MyThread1 t1=**new** MyThread1(obj);

MyThread2 t2=**new** MyThread2(obj);

t1.start();

t2.start();

}

}

Example 2 : with synchronization

**Syntax to use synchronized block**

**synchronized** (object reference expression) {

  //code block

}

Example :

**class** Table{

**void** printTable(**int** n){

**synchronized**(**this**){//synchronized block

**for**(**int** i=1;i<=5;i++){

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

**try**{

Thread.*sleep*(400);

}**catch**(Exception e){System.***out***.println(e);}

}

}

}//end of the method

}

**class** MyThread1 **extends** Thread{

Table t;

MyThread1(Table t){

**this**.t=t;

}

**public** **void** run(){

t.printTable(5);

}

}

**class** MyThread2 **extends** Thread{

Table t;

MyThread2(Table t){

**this**.t=t;

}

**public** **void** run(){

t.printTable(100);

}

}

**public** **class** TestSynchronizedBlock1{

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

Table obj = **new** Table();//only one object

MyThread1 t1=**new** MyThread1(obj);

MyThread2 t2=**new** MyThread2(obj);

t1.start();

t2.start();

}

}

3 Static Synchronization

Example :

**class** Table{

**synchronized** **static** **void** printTable(**int** n){

**for**(**int** i=1;i<=10;i++){

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

**try**{

Thread.*sleep*(400);

}**catch**(Exception e){}

}

}

}

**class** MyThread1 **extends** Thread{

**public** **void** run(){

Table.*printTable*(1);

}

}

**class** MyThread2 **extends** Thread{

**public** **void** run(){

Table.*printTable*(10);

}

}

**class** MyThread3 **extends** Thread{

**public** **void** run(){

Table.*printTable*(100);

}

}

**class** MyThread4 **extends** Thread{

**public** **void** run(){

Table.*printTable*(1000);

}

}

**public** **class** TestSynchronization4{

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

MyThread1 t1=**new** MyThread1();

MyThread2 t2=**new** MyThread2();

MyThread3 t3=**new** MyThread3();

MyThread4 t4=**new** MyThread4();

t1.start();

t2.start();

t3.start();

t4.start();

}

}

8 Collections :

ArrayList example :

package List;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Iterator;

public class ArrayL {

public static void main(String[] args) {

ArrayList<String> s1=new ArrayList<String>();

s1.add("sai");

s1.add("bhavna");

s1.add("raja");

s1.add("pawan");

s1.add("krithi");

s1.remove(2);

Iterator i1 = s1.iterator();

while(i1.hasNext()) {

System.out.println(i1.next());

}

Collections.sort(s1);

System.out.println(s1);

}

}

collections assignment

1:program 1:

package collassign;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Iterator;

import java.util.ListIterator;

public class Arraylist {

public static void main(String[] args) {

//all datatypes

ArrayList list =new ArrayList();

list.add("sai");

list.add(22);

list.add("developer");

list.add("B tech");

list.add(2, "frondend");

list.set(1, "ram");

System.out.println(list);

//foreach method

for(Object obj:list) {

System.out.println(list);

}

//iterator

Iterator i1=list.iterator();

while(i1.hasNext()) {

System.out.println(i1.next());

}

//listiterator

ListIterator i2=list.listIterator(list.size());

while(i2.hasPrevious()) {

System.out.println(i2.previous());

}

//integertype

ArrayList list1 =new ArrayList();

list1.add(99);

list1.add(22);

list1.add(32);

list1.add(65);

list1.add(2, 55);

list1.set(1, 56);

System.out.println(list1);

Collections.sort(list1);

System.out.println(list1);

Collections.reverse(list1);

System.out.println(list1);

Collections.shuffle(list1);

System.out.println(list1);

ArrayList<String> list2 =new ArrayList<String>();

list2.add("sai");

list2.add("ram");

list2.add("chandra");

list2.add("Btech");

list2.add(2, "nriit");

list2.set(1, "college");

System.out.println(list2);

Collections.sort(list2);

System.out.println(list2);

Collections.reverse(list2);

System.out.println(list2);

Collections.shuffle(list2);

System.out.println(list2);

ArrayList<String> list3 =new ArrayList<String>();

list3.add("Araveti");

list2.addAll(0, list3);

System.out.println("adding addall method : "+list2);

}

}

===========

program 2:

package collassign;

public class Friends {

String Frnd1;

public Friends(String frnd1) {

super();

Frnd1 = frnd1;

}

public String getFrnd1() {

return Frnd1;

}

public void setFrnd1(String frnd1) {

Frnd1 = frnd1;

}

@Override

public String toString() {

return "Friends [Frnd1=" + Frnd1 + "]";

}

}

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

extends

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

package collassign;

import java.util.LinkedList;

import java.util.Scanner;

public class ArrayLiUs {

public static void main(String[] args) {

@SuppressWarnings("resource")

Scanner sc=new Scanner(System.in);

LinkedList<Friends> ls=new LinkedList<Friends>();

String res=null;

do {

System.out.println("enter the id and name and percentage");

String Frnd1=sc.next();

Friends frnd=new Friends(Frnd1);

boolean rs=ls.add(frnd);

if(rs) {

System.out.println("added data");

}

else {

System.out.println("not added data");

}

System.out.println("do you have more frnds give 'yes' or 'no'");

res=sc.next();

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

System.out.println("First Frnd details :"+ls.getFirst());

System.out.println("last Frnd details :"+ls.getLast());

// all frnds

for(int i=1;i<=ls.size();i++) {

System.out.println(ls);

}

}

}

====================

program 3:

package collassign;

import java.util.\*;

public class Linked {

public static void main(String [] args)

{

LinkedList<String> ll=new LinkedList<String>();

ll.add("Ravi");

ll.add("Vijay");

ll.add("Ajay");

ll.add("Anuj");

ll.add("Gaurav");

ll.add("Harsh");

ll.add("Virat");

ll.add("Gaurav");

ll.add("Harsh");

ll.add("Amit");

System.out.println("Initial list of elements: "+ll);

//Removing specific element from arraylist

ll.remove("Vijay");

System.out.println("After invoking remove(object) method: "+ll);

//Removing element on the basis of specific position

ll.remove(0);

System.out.println("After invoking remove(index) method: "+ll);

LinkedList<String> ll2=new LinkedList<String>();

ll2.add("Ravi");

ll2.add("Hanumat");

// Adding new elements to arraylist

ll.addAll(ll2);

System.out.println("Updated list : "+ll);

//Removing all the new elements from arraylist

ll.removeAll(ll2);

System.out.println("After invoking removeAll() method: "+ll);

//Removing first element from the list

ll.removeFirst();

System.out.println("After invoking removeFirst() method: "+ll);

//Removing first element from the list

ll.removeLast();

System.out.println("After invoking removeLast() method: "+ll);

//Removing first occurrence of element from the list

ll.removeFirstOccurrence("Gaurav");

System.out.println("After invoking removeFirstOccurrence() method: "+ll);

//Removing last occurrence of element from the list

ll.removeLastOccurrence("Harsh");

System.out.println("After invoking removeLastOccurrence() method: "+ll);

//Removing all the elements available in the list

ll.clear();

System.out.println("After invoking clear() method: "+ll);

}

}

==============================

program 4:

package collassign;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Iterator;

import java.util.LinkedList;

import java.util.ListIterator;

public class Linkedlist {

public static void main(String[] args) {

//all datatypes

LinkedList list =new LinkedList();

list.add("sai");

list.add(22);

list.add("developer");

list.add("B tech");

list.add(2, "frondend");

list.set(1, "ram");

System.out.println(list);

//foreach method

for(Object obj:list) {

System.out.println(list);

}

//iterator

Iterator i1=list.iterator();

while(i1.hasNext()) {

System.out.println(i1.next());

}

//listiterator

ListIterator i2=list.listIterator(list.size());

while(i2.hasPrevious()) {

System.out.println(i2.previous());

}

//integertype

LinkedList list1 =new LinkedList();

list1.add(99);

list1.add(22);

list1.add(32);

list1.add(65);

list1.add(2, 55);

list1.set(1, 56);

System.out.println(list1);

Collections.sort(list1);

System.out.println(list1);

Collections.reverse(list1);

System.out.println(list1);

Collections.shuffle(list1);

System.out.println(list1);

LinkedList<String> list2 =new LinkedList<String>();

list2.add("sai");

list2.add("ram");

list2.add("chandra");

list2.add("Btech");

list2.add(2, "nriit");

list2.set(1, "college");

System.out.println(list2);

Collections.sort(list2);

System.out.println(list2);

Collections.reverse(list2);

System.out.println(list2);

Collections.shuffle(list2);

System.out.println(list2);

}

}

==============================

program5: user input:

package collassign;

public class Student {

String name;

double per;

int id;

public Student(String name,double per,int id) {

// TODO Auto-generated constructor stub

this.name=name;

this.per=per;

this.id=id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public double getPer() {

return per;

}

public void setPer(double per) {

this.per = per;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

@Override

public String toString() {

return "Student [name=" + name + ", id=" + id + ", perc=" + per + "]";

}

}

----------

extends

----------

package collassign;

import java.util.LinkedList;

import java.util.Scanner;

public class School {

public static void main(String[] args) {

@SuppressWarnings("resource")

Scanner sc=new Scanner(System.in);

LinkedList<Student> ls=new LinkedList<Student>();

String res=null;

do {

System.out.println("enter the id and name and percentage");

String name=sc.next();

int id=sc.nextInt();

double per=sc.nextDouble();

Student std=new Student(name,per,id);

boolean rs=ls.add(std);

if(rs) {

System.out.println("added data");

}

else {

System.out.println("not added data");

}

System.out.println("do you have more students");

res=sc.next();

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

System.out.println("First student details :"+ls.getFirst());

System.out.println("last student details :"+ls.getLast());

// all student details

for(int i=1;i<=ls.size();i++) {

System.out.println(ls);

}

}

}

9 Lamda Expressions:

Example 1 :

**package** LamdaExpsn;

**public** **class** LamdaE {

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

Thread t1=**new** Thread(() ->{

**for**(**int** i=1;i<=5;i++) {

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

**try** {

Thread.*sleep*(200);

} **catch** (Exception e) {

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

}

}

});

Thread t2=**new** Thread(() ->{

**for**(**int** i=1;i<=5;i++) {

System.***out***.println("i am a lamda expression");

**try** {

Thread.*sleep*(200);

} **catch** (Exception e) {

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

}

}

});

Thread t3=**new** Thread(() ->{

**for**(**int** i=1;i<=5;i++) {

System.***out***.println("i am going");

**try** {

Thread.*sleep*(200);

} **catch** (Exception e) {

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

}

}

});

t1.start();

t2.start();

t3.start();

}

}

Lamda with runnable Example :

**package** Runnable;

**public** **class** LamdaWithRun {

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

Runnable r1=**new** Runnable(){

@Override

**public** **void** run() {

**for**(**int** i=1;i<=10;i++) {

System.***out***.println("hiii i am runnable 1 with lamdafunction");

**try** {

Thread.*sleep*(200);

} **catch** (Exception e) {

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

}

}

}};

Runnable r2= ()->{

**for**(**int** i=1;i<=10;i++) {

System.***out***.println("hiii i am runnable 2 with lamdafunction");

**try** {

Thread.*sleep*(200);

} **catch** (Exception e) {

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

}

}};

Thread t1=**new** Thread(r1);

Thread t2=**new** Thread(r2);

t1.start();

t2.start();

}

}

3.By usong array with lamda expression

import java.util.Arrays;

import java.util.List;

import java.util.function.Consumer;

public class Arralist {

public static void main(String[] args)

{

List<Integer> list

= Arrays.asList(99,98,97,96,95,94,55);

list.forEach(number

-> System.out.print(

number + " "));

}

}

10 Jdbc topic :

Example : jdbc

jdbc crud operation:

program1: creating table:

package AssIgn;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.Statement;

public class Empl {

private static final String emp="CREATE TABLE EMPLOYEE1( "

+ "EID INT NOT NULL,"

+ "ENAME VARCHAR(20) NOT NULL,"

+ "EDESI VARCHAR(20) NOT NULL,"

+ "EAGE INT not null,"

+ "ESALARY INT not null,"

+ "PRIMARY KEY (EID))";

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

String url="jdbc:mysql://localhost:3306/demovirtusa";

String user="root";

String pass="Sairamchandra65@";

try {

//connection

Connection conn=DriverManager.getConnection(url,user,pass);

//statement

Statement st=conn.createStatement();

System.out.println("connected");

st.executeUpdate(emp);

System.out.println("table created");

conn.close();

} catch (Exception e) {

System.out.println(e);

}

}

}

=========================================

program table inserting values data:

package AssIgn;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.Statement;

public class Insert {

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

String url="jdbc:mysql://localhost:3306/demovirtusa";

String user="root";

String pass="Sairamchandra65@";

try {

//connection

Connection conn=DriverManager.getConnection(url,user,pass);

//statement

Statement st=conn.createStatement();

String st1="INSERT INTO EMPLOYEE1 VALUES(95,'ramesh','SR\_ARCHI',60,170000)";

st.executeUpdate(st1);

// st1="INSERT INTO EMPLOYEE1 VALUES(2,'bhavna','trainer',22,50000)";

// st.executeUpdate(st1);

// st1="INSERT INTO EMPLOYEE1 VALUES(3,'pallavi','SR\_ARCHI',22,65000)";

// st.executeUpdate(st1);

// st1="INSERT INTO EMPLOYEE1 VALUES(4,'harsha','jun\_archi',24,56000)";

// st.executeUpdate(st1);

// st1="INSERT INTO EMPLOYEE1 VALUES(5,'sai','devel',25,35000)";

// st.executeUpdate(st1);

System.out.println("inserted all details");

int rows=st.executeUpdate(st1);

if (rows>0) {

System.out.println("is inserted perfect");

} else {

System.out.println("something error");

}

conn.close();

} catch (Exception e) {

System.out.println(e);

}

}

}

====================================================

program age greaterthan age 40 :

package AssIgn;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class Age {

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

String url="jdbc:mysql://localhost:3306/demovirtusa";

String user="root";

String pass="Sairamchandra65@";

try {

//connection

Connection conn=DriverManager.getConnection(url,user,pass);

//statement

System.out.println("yaa i got connection");

String st1="Update employee1 SET edesi=? where eage>?";

PreparedStatement pre=conn.prepareStatement(st1);

pre.setString(1, "manager");

pre.setLong(2, 40);

int rows=pre.executeUpdate();

if (rows>0) {

System.out.println("updated");

} else {

System.out.println("not updated");

}

conn.close();

} catch (Exception e) {

System.out.println(e);

}

}

}