**Java: Day2**

\*\*Create classes under multiple packages: Calling classes under different packages

package one;

import static java.lang.System.\*;

import java.util.\*;

public class Employee

{

    private String name;

    private double salary;

    private Date hireday;

    public Employee(String s,double s1,int year,int month,int day)

    {

    name=s;

    salary=s1;

    GregorianCalendar calendar=new GregorianCalendar(year,month-1,day);

    hireday=calendar.getTime();

    }

    public String getName()

    {

        return name;

    }

    public boolean equals(Object other)

    {

        if(this!=other)

        return false;

        if(other==null)

        return false;

        if(getClass()!=other.getClass())

        return false;

        Employee e1=(Employee)other;

        return name.equals(e1.name)&&salary==e1.salary;

    }

}

package one;

public class Manager extends Employee

{

    public Manager(String s,double s1,int year,int month,int day)

    {

        super(s,s1,year,month,day);

    }

}

import static java.lang.System.\*;

import one.\*;

class EqualTest

{

    public static void main(String[] args)

    {

        one.Employee alice1=new one.Employee("alice",4500.00,1987,11,23);

        one.Employee alice2=alice1;

        one.Employee alice3=new one.Employee("alice",4500.0,1987,11,23);

        out.println("alice1==alice2"+alice1.equals(alice2));

        out.println("alice1==alice3"+alice1.equals(alice3));

        one.Manager boss=new one.Manager("alice",4500.0,1987,11,23);

        alice2=boss;

        out.println("alice1==alice2"+alice1.equals(alice2));

    }

}

\*\*Try/catch/Finally

|  |
| --- |
| public class exceptions{ |

|  |  |
| --- | --- |
|  | public static void main(String Args[]){ |

|  |  |
| --- | --- |
|  | int[] array = new int[3]; |

|  |  |
| --- | --- |
|  | try{ |

|  |  |
| --- | --- |
|  | for(int i=0;i<4;++i){ |

|  |  |
| --- | --- |
|  | array[i] = i; |

|  |  |
| --- | --- |
|  | } |

|  |  |
| --- | --- |
|  | System.out.println(array); |

|  |  |
| --- | --- |
|  | } |

|  |  |
| --- | --- |
|  | catch(ArrayIndexOutOfBoundsException e){ |

|  |  |
| --- | --- |
|  | //printed just to inform that we have entered the catch block |

|  |  |
| --- | --- |
|  | System.out.println("Exception occured"); |

|  |  |
| --- | --- |
|  | } |

|  |  |
| --- | --- |
|  | finally{ |

|  |  |
| --- | --- |
|  | System.out.println(array); |

|  |  |
| --- | --- |
|  | //method call to continue program |

|  |  |
| --- | --- |
|  | } |

|  |  |
| --- | --- |
|  | } |

|  |  |
| --- | --- |
|  | } |

\*\*Final keyword

It can be used in different contexts such as variable, method, class.

**Final variable**: We cannot change the value of the variable.

class Bike9{

 final int speedlimit=90;//final variable

void run(){

  speedlimit=400;

 }

public static void main(String args[]){

 Bike9 obj=new  Bike9();

 obj.run();

 }

}

**Final Method:** You cannot override a final method but can inherit it.

class Bike{

  final void run(){System.out.println("running");}

}

class Honda extends Bike{

   void run(){System.out.println("running safely with 100kmph");}

   public static void main(String args[]){

   Honda honda= new Honda();

   honda.run();

   }

}

**Final Class**: A final class cannot be extended.

final class Bike{}

class Honda1 extends Bike{

  void run(){System.out.println("running safely with 100kmph");}

  public static void main(String args[]){

  Honda1 honda= new Honda();

  honda.run();

  }  }

\*\* Implement an interface with a class implementing interface

Interface MyInterface

{

Public void method1();

Public void method2();

}

Class XYZ inmplements MyInterface

{

Public void method1()

{

System.out.println(“implementation of method1”);

}

Public void method2()

{

System.out.println(“implementation of method2”);

}

Public static void main(String args[])

{

MyInterface obj=new XYZ();

Obj.method1;

Obj.method2;

}

}

\*\*Abstract Class and methods

// An abstract class with constructor

abstract class Base {

    Base() { System.out.println("Base Constructor Called"); }

    abstract void fun();

}

class Derived extends Base {

    Derived() { System.out.println("Derived Constructor Called"); }

    void fun() { System.out.println("Derived fun() called"); }

}

class Main {

    public static void main(String args[]) {

       Derived d = new Derived();

    }

}

\*\*Overloading

class Calculation{

  void sum(int a,int b){System.out.println(a+b);}

  void sum(int a,int b,int c){System.out.println(a+b+c);}

  public static void main(String args[]){

  Calculation obj=new Calculation();

  obj.sum(10,10,10);

  obj.sum(20,20);

  }  }

\*\*Overriding

public int getPerimeter()

{

return 2 \* ( height + width ) ;

}

public int getPerimeter()

{

return 2 \* ( super.getHeight() + super.getWidth() + 4 ) ;

}

public static void main( String [] args )

{

Rectangle r = new Rectangle( 3, 5 ) ;

FramedRectangle f = new FramedRectangle( 3, 5 ) ;

System.out.println( "r's perimeter is: " + r.getPerimeter() ) ;

System.out.println( "f's perimeter is: " + f.getPerimeter() ) ;

}

\*\*Polymorphism

Polymorphism is a method having more than one form.It can be defined as

Overloading (or) Compile time polymorphism.

Overriding (or) Runtime polymorphism.

Write code to save data into excel file and read from excel file (poi and jexcel api)

How to write update data into xml file and read data from xml file

\*\*How to add elements ,retrieve elements into int, str array list/array

Package demo

Import java.io.\*;

Import java.util.\*;

Public static void main(String args[])

{

ArrayList<Integer> a=new ArrayList<Integer>(5);

a.add(15);

a.add(20);

a.add(45);

a.add(34);

a.add(2,56);//adding at position 3.

For(Integer number:a)

System.out.println(“Number:”:+number);

Int p=a.get(0);//to retrieve element

}

}

}

Package demo

Import java.io.\*;

Import java.util.\*;

Public static void main(String args[])

{

ArrayList<String> a=new ArrayList<String>(5);

a.add(15);

a.add(20);

a.add(45);

a.add(34);

a.add(2,56);//adding at position 3.

For(String number:a)

System.out.println(“Number:”:+String);

String p=a.get(0);//to retrieve element

}

}

}

\*\*How to add and retrieve elements to a hash map

public class MapDemo

{

    public static void main(String args[])

    {

        Map map = new HashMap();

        //Adding values to the HashMap

        map.put("test key 1", "test value 1");

        map.put("test key 2", "test value 2");

        map.put("test key 3", "test value 3");

        System.out.println("Retrieving values from HashMap");

        retrieveValuesFromListMethod(map);

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

    }

    /\*This method retrieves values from Map

     \*/

    public static void retrieveValuesFromListMethod(Map map)

    {

        Set keys = map.keySet();

        Iterator itr = keys.iterator();

        String key;

        String value;

        while(itr.hasNext())

        {

            key = (String)itr.next();

            value = (String)map.get(key);

            System.out.println(key + " - "+ value);

        }

    }

}

\*\*How to retrieve and add elements to hashset

import java.util.Collections;

import java.util.Enumeration;

import java.util.HashSet;

import java.util.Iterator;

import java.util.Set;

public class SetDemo

{

    public static void main(String args[])

    {

        Set set = new HashSet();

        //Adding values to the HashSet

        set.add("test1");

        set.add("test2");

        set.add("test3");

        System.out.println("Retrieving values from HashSet using Iterator");

        retrieveValuesFromListMethod1(set);

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

        System.out.println("Retrieving values from HashSet using Enumeration");

        retrieveValuesFromListMethod2(set);

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

    }

    /\*This method retrieves values from HashSet using Iterator

     \*/

    public static void retrieveValuesFromListMethod1(Set set)

    {

        Iterator itr = set.iterator();

        while(itr.hasNext())

        {

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

        }

    }

/\*This method retrieves values from HashSet using Enumeration

     \*/

    public static void retrieveValuesFromListMethod2(Set set)

    {

        Enumeration e = Collections.enumeration(set);

        while(e.hasMoreElements())

        {

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

        }

    }

}

write code to connect to JDBC to get rows from employee table

Write method to return list of rows code to loop through

\*\* create Employee class Add employee class to list collection create method that return list of employee collection

public class Employee {

int empid;

String name;

int age;

public Employee(int empid,String name,int age)

{

this.empid=empid;

this.name=name;

this.age=age;

}

public int getEmpid() {

return empid;

}

public void setEmpid(int empid) {

this.empid = empid;

}

public String getname() {

return name;

}

public void setname(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

comparator class:

public class Employee\_comparator implements Comparator<Employee> {

@Override

public int compare(Employee object1, Employee object2) {

return object1.getname().compareTo(object2.getname());

}

}

main class:

import java.util.ArrayList;

import java.util.Collections;

import java.util.Iterator;

import java.util.List;

public class Employee\_Main {

/\*\*

\* @param args

\*/

public static void main(String[] args) {

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

list.add(new Employee(33186,"varun",23));

list.add(new Employee(33187,"deepak",23));

list.add(new Employee(33188,"apple",23));

list.add(new Employee(33189,"rohan",23));

Collections.sort(list,new Employee\_comparator());

for(int i=0;i<list.size();i++){

System.out.print("age:"+list.get(i).getAge());

System.out.print("empid:"+list.get(i).getEmpid());

System.out.println("name:"+list.get(i).getname());

}

Iterator<Employee> itr=list.iterator();

while(itr.hasNext())

{

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

}

}

}

\*\*Difference between string,string builder,stringbuffer

String: threadsafe, synchronized, immutable

String buffer: threadsafe, synchronized, mutable, faster than string

String Builder: threadsafe, unsynchronized, mutable, faster than string and string buffer

public class ConcatTest{

    public static void main(String[] args){

        long startTime = System.currentTimeMillis();

        StringBuffer sb = new StringBuffer("Java");

        for (int i=0; i<10000; i++){

            sb.append("Tpoint");

        }

        System.out.println("Time taken by StringBuffer: " + (System.currentTimeMillis() - startTime) + "ms");

        startTime = System.currentTimeMillis();

        StringBuilder sb2 = new StringBuilder("Java");

        for (int i=0; i<10000; i++){

            sb2.append("Tpoint");

        }

        System.out.println("Time taken by StringBuilder: " + (System.currentTimeMillis() - startTime) + "ms");

    }  }