Q:- What is wrapper class ? Use of Wrapper class ?

Ans :- A class which incapsulate types so that type can be used as object instance wherever required.

In situation, where we need to treat primitive data types as an object wrapper class can be used.

e.g. Integer for int etc.

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

Q:- What is operator and its requirement?

Ans :- Operators are special symbols that perform specific operations on one, two, or three operands

and then return a result.

Operator are used to perform any kind of operation like +,/,-,% etc.

It means it can be used to manupulate primitive data type.

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

Q:- What is condition construct and why it is required ?

Ans :- Conditional construct is a statement which check for any condition and execute some part of code depending whether

condition is true or false. In Java, there are two conditional construct, if-else and switch statement.

It can be used where any decision has to be taken depending upon certion condition.

The if-else statement allows us to select between two alternatives.

The switch statement to choose between multiple alternatives.

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

Q:- Difference between if-else if and switch case ?

Ans :- If statement is used to select among two alternatives.

It uses a boolean expression to decide which alternative should be executed.

The switch statement is used to select among multiple alternatives because it uses an int expression

to determine which alternatives need be executed.

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

Q:- Diffrent phase of looping ?

Ans :- While loop, for loop and Do-while

While loop -

while(Boolean\_expression) {

    // Statements

}

Repeat execution until condition is true.

For loop -

for(initialization; Boolean\_expression; update) {

  // Statements

}

Execute statements for no of time given in loop.

Do-while loop -

do {

    // Statements

}while(Boolean\_expression);

Where statements need to be execute at least once and then check for condition for repeatation.

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

Q:- WAP using array to accept 10 numbers and display it in ascending number.

package javaapplication1;

import java.util.Arrays;

import java.util.Scanner;

/\*\*

 \*

 \* @author DUVISHAL

 \*/

public class JavaApplication1 {

    int[] arrayInt = new int[10];

    void accpetArray()

    {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the array element :- ");

        for(int i=0; i<10; i++)

        {

            System.out.print(i+1 +" Element :- ");

            arrayInt[i]= sc.nextInt();

            System.out.println();

        }

    }

    void displayArray()

    {

        for (int number : arrayInt) {

            System.out.print(number + " ");

        }

        System.err.println("");

    }

    void sortArray()

    {

        int temp = 0;

         for(int i=0; i < 10; i++){

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

                if(arrayInt[j-1] > arrayInt[j]){

                       temp = arrayInt[j-1];

                       arrayInt[j-1] = arrayInt[j];

                       arrayInt[j] = temp;

               }

            }

         }

    }

    public static void main(String[] args) {

        // TODO code application logic here

        JavaApplication1 obj = new JavaApplication1();

        obj.accpetArray();

        System.err.println("Array before sorting.....");

        obj.displayArray();

        System.err.println("Array after sorting.....");

        obj.sortArray();

        obj.displayArray();

    }

}

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

Q:- WAP to accpet 10 numbers using array and display the sum and average.

package javaapplication1;

import java.util.Arrays;

import java.util.Scanner;

/\*\*

 \*

 \* @author DUVISHAL

 \*/

public class JavaApplication1 {

    int[] arrayInt = new int[10];

    void accpetArray()

    {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the array element :- ");

        for(int i=0; i<10; i++)

        {

            System.out.print(i+1 +" Element :- ");

            arrayInt[i]= sc.nextInt();

            System.out.println();

        }

    }

    void displayArray()

    {

        for (int number : arrayInt) {

            System.out.print(number + " ");

        }

        System.err.println("");

    }

    void sumAverage()

    {

        int temp = 0;

        for(int x: arrayInt)

            temp+=x;

        System.err.println("Sum of Array :- " + temp);

        System.err.println("Average of Array :- " + ((float)temp/10));

    }

    public static void main(String[] args) {

        // TODO code application logic here

        JavaApplication1 obj = new JavaApplication1();

        obj.accpetArray();

        System.err.println("Array before sorting.....");

        obj.displayArray();

        obj.sumAverage();

    }

}

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

Q:- WAP to accpet marks of 10 students using array and find name of highest scorer.

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

package javaapplication1;

import java.util.Arrays;

import java.util.Scanner;

/\*\*

 \*

 \* @author DUVISHAL

 \*/

public class JavaApplication1 {

    String[][] arrayStudent = new String[2][10];

    void accpetArray()

    {

        Scanner sc = new Scanner(System.in);

        for(int i=0; i<10; i++)

        {

            System.out.print("Enter the name of Student = ");

            arrayStudent[0][i]= sc.next();

             System.out.println();

            System.out.print("Enter the marks = ");

            arrayStudent[1][i]= sc.next();

            System.out.println();

        }

    }

    void displayArray()

    {

        for (int i=0; i<10; i++) {

             System.out.println("Name of Student = " + arrayStudent[0][i] + " and marks = " + arrayStudent[1][i] );

        }

        System.err.println("");

    }

    void findMaxMarks()

    {

        int temp = Integer.parseInt(arrayStudent[1][1]);

        int index = 0;

        for(int i=1; i<10; i++)

        {

            if(Integer.parseInt(arrayStudent[1][i]) > temp)

                {

                    temp=Integer.parseInt(arrayStudent[1][i]);

                    index = i ;

                }

        }

        System.err.println("Student with highest marks = " + arrayStudent[0][index] + " and marks = " +arrayStudent[1][index]);

    }

    public static void main(String[] args) {

        // TODO code application logic here

        JavaApplication1 obj = new JavaApplication1();

        obj.accpetArray();

        System.err.println("Array before sorting.....");

        obj.displayArray();

        obj.findMaxMarks();

    }

}

Q: What is function return type ?

Ans :- Function return type is type of data mentioned in method signature, going to be returned by the function after

executing to caller function.

Q: What is the use of passing parameter inside the function?

Ans :- Allow the function to use parameter which is out of scope of function.

Q: What is void? Why it is used ?

Ans :- Void is the Java keyword that tells the compiler that a function will not be returning any value after it is executed.

Q: What is use of String[] aa in main function?

Ans :- String[] aa is array of String which is used to collect argument (command line arguement) during execution of at start which may help program

for  initialization. It is not mandatory that we need to pass some argument while executing the program at start.

Q: Program for addition, substraction and Multiplication of two 3\*3 Matrix.

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

package javaapplication1;

import java.util.Arrays;

import java.util.Scanner;

/\*\*

 \*

 \* @author DUVISHAL

 \*/

public class JavaApplication1 {

    int[][] firstMatrix = new int[3][3];

    int[][] secondMatrix = new int[3][3];

    int[][] productMatrix = new int[3][3];

    String[][] arrayStudent = new String[2][10];

    void accpetMatrix()

    {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the elements of first martix - \n");

        for (int i = 0; i < 3; i++)

        {

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

            {

                System.err.print("Element [" + i +"][" + j +"]");

                firstMatrix[i][j]=sc.nextInt();

                System.err.println();

            }

        }

        System.out.println("Enter the elements of second martix - \n");

        for (int i = 0; i < 3; i++)

        {

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

            {

                System.err.print("Element [" + i +"][" + j +"]");

                secondMatrix[i][j]=sc.nextInt();

                System.err.println();

            }

        }

    }

    void displayMatrix()

    {

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

        for (int i = 0; i < 3; i++)

        {

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

            {

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

            }

            System.out.println();

        }

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

        for (int i = 0; i < 3; i++)

        {

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

            {

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

            }

            System.out.println();

        }

    }

    void matrixProduct()

    {

        System.out.println("Product of  Matrix - ");

        for (int i = 0; i < 3; i++)

        {

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

             {

                 for (int k = 0; k < 3; k++)

                 {

                     productMatrix[i][j] = productMatrix[i][j] + firstMatrix[i][k] \* secondMatrix[k][j];

                 }

             }

        }

        for (int i = 0; i < 3; i++)

        {

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

            {

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

            }

            System.out.println();

        }

    }

    void matrixAddition()

    {

        System.out.println("Addition of  Matrix - ");

        for (int i = 0; i < 3; i++)

        {

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

             {

                 productMatrix[i][j] = firstMatrix[i][j] + secondMatrix[i][j];

             }

        }

        for (int i = 0; i < 3; i++)

        {

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

            {

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

            }

            System.out.println();

        }

    }

    void matrixSubstraction()

    {

        System.out.println("Substraction of  Matrix - ");

        for (int i = 0; i < 3; i++)

        {

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

             {

                 productMatrix[i][j] = firstMatrix[i][j] - secondMatrix[i][j];

             }

        }

        for (int i = 0; i < 3; i++)

        {

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

            {

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

            }

            System.out.println();

        }

    }

    public static void main(String[] args) {

        // TODO code application logic here

        JavaApplication1 obj = new JavaApplication1();

        obj.accpetMatrix();

        obj.displayMatrix();

        obj.matrixProduct();

        obj.matrixAddition();

        obj.matrixSubstraction();

    }

}

Q:- What is constructor?

Ans :- A constructor is a special method that is used to initialize an object.

Every class has a constructor,if we don't explicitly declare a constructor for any java class the compiler builds a default constructor for that class.

A constructor does not have any return type.

The name of the constructor must be the same as the name of the class.

Q:- What is parameterized constructor ?

Ans :- A constructor with parameter is known as parameterized constructor.

class Student

{

    int id;

    String name;

    Student4(int i,String n)

    {

    id = i;

    name = n;

    }

    void display()

    {

System.out.println(id+" "+name);

    }

    public static void main(String args[])

   {

    Student s1 = new Student4(123,"Vishal");

    s1.display();

   }

}

Q:- Rules to follow constructor ?

Ans :-

Constructors have same name as the class name.

Constructors have a parameter list like methods but don’t have a return type, nor even void.

Q:- What is overloading ?

Ans -: Method Overloading is a feature that allows a class to have two or more methods having same name,

if their argument lists are different.

It is required in case we want that same method behave diffrently by providing diffrent attribute.

public class Data{

    ...

    public void draw(String s) {

        ...

    }

    public void draw(int i) {

        ...

    }

    public void draw(double f) {

        ...

    }

    public void draw(int i, double f) {

        ...

    }

}

Q:- What is overriding?

Ans -: Method overriding allows a subclass or child class to provide a specific implementation of a method

that is already provided by one of its superclasses or parent classes.

It is required in case child class want to improve/modify bahaviour of any method provide by it's parent class.

class Animal

{

 public void eat()

 {

  System.out.println("Generic Animal eating");

 }

}

class Dog extends Animal

{

 public void eat()

 {

  System.out.println("Dog eat meat");

 }

}

Q:- What is inheritance ? How it can be implemented ?

Ans :- Inheritance in java is a mechanism in which one object acquires all the properties and

behaviors of parent object.

class Vehicle

{

    String vehicleType;

}

public class Car extends Vehicle {

    String modelType;

    public void showDetail()

    {

        vehicleType = "Car";        //accessing Vehicle class member

        modelType = "sports";

        System.out.println(modelType+" "+vehicleType);

    }

    public static void main(String[] args)

    {

        Car car =new Car();

        car.showDetail();

    }

}

Q:- What is interface and how it is diffrent from Abstract class ?

Ans :-  Interface is a reference type, similar to a class, that can contain only constants,

method signatures, default methods, static methods, and nested types.

Interfaces cannot be instantiated—they can only be implemented by classes or extended by other interfaces.

Interface is way to achieve 100% abstraction while Abstract class may/may not provide 100% abstraction.

Because in abstract class you can provide method body to non-abstract method.

Q:- WAP in java to create userdefined exceeption which allow to perform bank transaction

1. If user try to withdraw amount > balance must throw isufficeint fund exception.

2. Before withdrwa, ask for valid pin, if not valid then throw invalid pin exception.

3. Ask foe PAN details if deposit > 20000, if not valid then throw invalid transaction.

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

package javaapplication1;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

/\*\*

 \*

 \* @author DUVISHAL

 \*/

class bankException extends Exception

{

    public bankException(String s) {

        super(s);

    }

}

class banking

{

    private String name;

    private float amount;

    private int pin;

    private String PAN;

    public banking(String name, float amount, int pin, String PAN) {

        this.name = name;

        this.amount = amount;

        this.pin = pin;

        this.PAN = PAN;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public float getAmount() {

        return amount;

    }

    public void setAmount(float amount) {

        this.amount = amount;

    }

    public int getPin() {

        return pin;

    }

    public void setPin(int pin) {

        this.pin = pin;

    }

    public String getPAN() {

        return PAN;

    }

    public void setPAN(String PAN) {

        this.PAN = PAN;

    }

    public void deposit(float amt) throws bankException,IOException

    {

        if(amt > 20000)

        {

            System.out.println("Please provide PAN number");

            String panDetails = (new BufferedReader(new InputStreamReader(System.in))).readLine();

            if(panDetails.equals(this.getPAN()))

                this.amount+=amt;

            else

                throw new bankException("PAN Details are not matching. Transaction declined.");

        }

        else

        {

            this.amount+=amt;

            System.err.println(amt + " has been added to your account and new balance is - " + this.amount);

        }

    }

    public void BalanceDisplay()

    {

        System.err.println("Your Balance is - " + this.getAmount());

    }

    public void withdrawl(float amt) throws bankException, IOException

    {

        System.err.println("Please enter your PIN");

        int pindetails = Integer.parseInt((new BufferedReader(new InputStreamReader(System.in))).readLine());

        if(this.pin == pindetails)

        {

            if((this.amount - amt ) > 0)

            {

                this.amount -=amt;

                System.err.println(amt + " has been withdrawn and new balance is " + this.amount);

            }

            else

                throw new bankException("Transaction failed...Insufficeint fund");

        }

        else

            throw  new bankException("Transaction declined. Incorrect PIN.");

    }

}

public class userDefinedException {

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

        banking customer1= new banking("Vishal", 0, 1111, "ABCDEFG");

        customer1.BalanceDisplay();

        try

        {

            customer1.deposit((float) 20000.0);

        }

        catch (bankException e)

        {

            System.err.println("Exception occured - " + e.getMessage());

        }

        try

        {

            customer1.withdrawl((float) 20000.0);

        }

        catch (bankException e)

        {

            System.err.println("Exception occured - " + e.getMessage());

        }

    }

}

Q:- What is Linked List ?

Ans :- LinkedList is a linked list(data structure) implementation of List interface.In addition to implementing the List interface,

the LinkedList class provides uniformly named methods to get, remove and insert an element at the beginning and end of the list.

These operations allow linked lists to be used as a stack, queue, or double-ended queue.

Q:- What is ListIterator?

Ans :- ListIterator Interface is used to traverse the element in backward and forward direction.

1. List Iterator you can move back word also while reading the elements.

2. With List Iterator you can obtain the index at any point while traversing.

3. In list iterator you can check previous and next elements.

4. With list iterator you can add new element at any point of time, while traversing.

5. With list iterator you can modify an element while traversing.

Q:- What are the characteristic of LinkedList ?

Ans :-

1. LinkedList internally uses doubly linked list to store the elements.

2. Manipulation with LinkedList is faster than ArrayList because it uses doubly linked list so no bit shifting is required in memory.

3. LinkedList class can act as a list and queue both because it implements List and Deque interfaces.

4. LinkedList is better for manipulating data.

Q:- Diffrence between Linkedlist and ArrayList ?

Ans :-

ArrayList:-

1. ArrayList internally uses dynamic array to store the elements.

2. Manipulation with ArrayList is slow because it internally uses array. If any element is removed from the array, all the bits are shifted in memory.

3. ArrayList class can act as a list only because it implements List only.

4. ArrayList is better for storing and accessing data.

5. In ArrayList, each index only holds the actual object(data). So memory overhead is less compared to Linkedlist

LinkedList:-

1. LinkedList internally uses doubly linked list to store the elements.

2. Manipulation with LinkedList is faster than ArrayList because it uses doubly linked list so no bit shifting is required in memory.

3. LinkedList class can act as a list and queue both because it implements List and Deque interfaces.

4. LinkedList is better for manipulating data.

5. Memory overhead in LinkedList is more as compared to ArrayList as node in LinkedList needs to maintain the addresses of next and previous node.

Q:- What is ArrayList ?

Ans :- Arraylist is a class which implements List interface and provides resizable array.

ArrayList supports dynamic arrays that can grow as needed.

Q:- What is Iterator ?

Ans :- Iterator in Java is used to traverse object inside a collection. Java Iterator allows you to easily access object stored in Collection.

Iterator is used for iterating (looping) various collection classes such as HashMap, ArrayList, LinkedList etc.

Q:- What are the characteristic of Arraylist?

Ans :-

1. Java ArrayList allows random access because array works at the index basis.

2. Java ArrayList class can contain duplicate elements.

3. Java ArrayList class is non synchronized.

4. Java ArrayList class maintains insertion order.

5. In Java ArrayList class, manipulation is slow because a lot of shifting needs to be occurred if any element is removed from the array list.

Q:- What are the advantage of using ArrayList?

Ans :-

1. You can define ArrayList as re-sizable array. Size of the ArrayList is not fixed. ArrayList can grow and shrink dynamically.

2. ArrayList class has many methods to manipulate the stored objects.

Q: What is Generic Class ?

Ans:- A class that can refer to any type is known as generic class.

Generics enable types (classes and interfaces) to be parameters when defining classes, interfaces and methods.

For ex. List<String> list = new ArrayList<String>();

Q:- What are the benefits o Generic Class?

Ans:-

1.Stronger type checks at compile time.

A Java compiler applies strong type checking to generic code and issues errors if the code violates type safety. Fixing compile-time errors is easier than fixing runtime errors, which can be difficult to find.

2.Elimination of casts.

3.Enabling programmers to implement generic algorithms.

By using generics, programmers can implement generic algorithms that work on collections of different types, can be customized, and are type safe and easier to read.

Q:- Use of Generic Class in real life scenario?

Ans:- If one has a List object (non-generic), one can store into it anything that can be cast into Object,

but there's no way of knowing at compile time what type of things one will get out of it.

By contrast, if one has a generic List<Animal>, the only things one can store into it are Animal

or derivatives thereof, and the compiler can know that the only things that will be pulled out of it will be Animal.

The compiler can thus allow things to be pulled out of the List and stored directly into fields of type Animal

without any need for run-time type checking.

Q:- What is Hashset?

Ans:- Java HashSet class is used to create a collection that uses a hash table for storage.

HashSet does not allow duplicate elements.

Q:- Characteristics of Hashset ?

Ans:-

1. Contains only unique element, duplicates are not allowed.

2. HashSet class uses HashMap internally to store the objects.

The keys of that HashMap object will be the elements of HashSet and their values will be a constant.

3. HashSet can have maximum one null element.

4. HashSet doesn’t maintain any order. The order of the elements will be largely unpredictable.

And it also doesn’t guarantee that order will remain constant over time

5. HashSet class is not synchronized. If you want synchronized HashSet, use Collections.synchronizedSet() method.

Q:- Adavantage of using Hashset?

Ans:-

1. It does not allow duplicate.

Q:- Diffrence between List and Set?

Ans:-

1) List is an ordered collection it maintains the insertion order, which means upon displaying the list content it will display the elements in the same order in which they got inserted into the list.

Set is an unordered collection, it doesn’t maintain any order.

2) List allows duplicates while Set doesn’t allow duplicate elements.

All the elements of a Set should be unique if you try to insert the duplicate element in Set it would replace the existing value.

3) List implementations: ArrayList, LinkedList etc.

Set implementations: HashSet, LinkedHashSet, TreeSet etc.

4) List allows any number of null values. Set can have only a single null value at most.

5) ListIterator can be used to traverse a List in both the directions(forward and backward) However it can not be used to traverse a Set. We can use Iterator (It works with List too) to traverse a Set.

6) List interface has one legacy class called Vector whereas Set interface does not have any legacy class.

Q:- WAP to maintain employee details using arraylist.

-> Accept minimum 5 employee details.

-> Display in proper order

-> Display name of employee having highest salary

-> Display details in the order of Salay.

Ans:-

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

package javaapplication1;

import java.util.Scanner;

import java.util.\*;

/\*\*

 \*

 \* @author DUVISHAL

 \*/

class Employee implements Comparable<Employee>

{

    private int id;

    private String name;

    private int sal;

    private int age;

    public Employee() {

    }

    public Employee(int id, String name, int sal, int age) {

        this.id = id;

        this.name = name;

        this.sal = sal;

        this.age = age;

    }

    public int getId() {

        return id;

    }

    public String getName() {

        return name;

    }

    public int getSal() {

        return sal;

    }

    public int getAge() {

        return age;

    }

    public void acceptDetails()

    {

        Scanner sc= new Scanner(System.in);

        System.err.print("Enter employee ID - ");

        id= sc.nextInt();

        System.out.println();

        System.err.print("Enter the Name of Employee - ");

        name = sc.next();

        System.out.println();

        System.err.print("Enter the Salary - ");

        sal=sc.nextInt();

        System.out.println();

        System.out.print("Enter the age of employee - )");

        age = sc.nextInt();

        System.out.println();

    }

    public void displyDetails()

    {

        System.err.println("Employee ID - " + id + ",  Employee Name - " + name + " , Age - " + age + ", Salary - " + sal);

    }

    @Override

    public int compareTo(Employee o) {

       return this.getSal()- o.getSal();

    }

    @Override

    public String toString()

    {

        return "Employee ID - " + id + ",  Employee Name - " + name + " , Age - " + age + ", Salary - " + sal + "\n";

    }

}

public class Arraylist

{

    public static void main(String[] args)

    {

        ArrayList<Employee> arraylist = new ArrayList<Employee>();

        int index=0, tempSal=0;

        Employee emp1= new Employee();

        Employee emp2= new Employee();

        Employee emp3= new Employee();

        Employee emp4= new Employee();

        Employee emp5= new Employee();

        emp1.acceptDetails();

        emp2.acceptDetails();

        emp3.acceptDetails();

        emp4.acceptDetails();

        emp5.acceptDetails();

        arraylist.add(emp1);

        arraylist.add(emp2);

        arraylist.add(emp3);

        arraylist.add(emp4);

        arraylist.add(emp5);

        // first way to find out max. sal employee

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

        {   if(arraylist.get(i).getSal() > tempSal)

                index = i;

        }

        System.err.println("Employee with Highest Sal -> " + arraylist.get(index));

        System.err.println("----------- Before Sorting -----------\n" + arraylist);

        Collections.sort(arraylist);

        System.err.println("----------- After Sorting -----------\n" + arraylist);

        // second way to find out max. sal employee

        System.err.println("Employee with Highest Sal -> " + arraylist.get(arraylist.size()-1));

        System.err.println("Employee with Lowet Sal -> " + arraylist.get(0));

    }

}

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

Q:- WAP in java to eliminate duplicate key in hashmap as user define objects?

Ans-

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

//hashmap for removing duplicate keys

package javaapplication1;

import java.util.Scanner;

import java.util.\*;

/\*\*

 \*

 \* @author DUVISHAL

 \*/

class Employee1

{

    private int id;

    private String name;

    private int sal;

    private int age;

    public Employee1() {

    }

    public Employee1(int id, String name, int sal, int age) {

        this.id = id;

        this.name = name;

        this.sal = sal;

        this.age = age;

    }

    public void displyDetails()

    {

        System. out.println("Employee ID - " + id + ",  Employee Name - " + name + " , Age - " + age + ", Salary - " + sal);

    }

    @Override

    public String toString()

    {

        return "Employee ID - " + id + ",  Employee Name - " + name + " , Age - " + age + ", Salary - " + sal + "\n";

    }

    @Override

    public int hashCode() {

        int hash = 7;

        hash = 83 \* hash + this.id;

        hash = 83 \* hash + Objects.hashCode(this.name);

        hash = 83 \* hash + this.sal;

        hash = 83 \* hash + this.age;

        return hash;

    }

    @Override

    public boolean equals(Object obj) {

        if (this == obj) {

            return true;

        }

        if (obj == null) {

            return false;

        }

        if (getClass() != obj.getClass()) {

            return false;

        }

        final Employee1 other = (Employee1) obj;

        if (this.id != other.id) {

            return false;

        }

        if (this.sal != other.sal) {

            return false;

        }

        if (this.age != other.age) {

            return false;

        }

        if (!Objects.equals(this.name, other.name)) {

            return false;

        }

        return true;

    }

}

public class hashMap

{

    public static void main(String[] args)

    {

        HashMap<Employee1,Integer> hashmap = new HashMap<Employee1, Integer>();

        hashmap.put(new Employee1(100,"Vishal", 20000,25),1);

        hashmap.put(new Employee1(101,"Mehul", 27000, 24),2);

        hashmap.put(new Employee1(107,"Vivek", 21000, 30),3);

        hashmap.put( new Employee1(109,"Shyam", 30000, 28),4);

        System.err.println("Hashmap Before duplicate Key");

         for(Map.Entry m:hashmap.entrySet())

         {

            System.out.print("Key - " + m.getKey()+ ", Value - " +m.getValue() );

             System. out.println();

         }

         hashmap.put(new Employee1(100,"Vishal", 20000,25),11);

         System.out.println("Hashmap after adding duplicate Key");

         for(Map.Entry m:hashmap.entrySet())

         {

            System.out.print("Key - " + m.getKey()+ ", Value - " +m.getValue() );

            System. out.println();

         }

    }

}

Q:- WAP in java to create generic class to accept employee age and salary.

Ans:-

import java.util.Scanner;

class genricEmployee<A,S>

{

private A age;

private S salary;

public genricEmployee() {

super();

}

public genricEmployee(A age, S salary) {

super();

this.age = age;

this.salary = salary;

}

public void acceptDisplay()

{

Scanner sc =new Scanner(System.in);

System.out.print("Enter the age of the employee - ");

age= (A)sc.next();

System.out.println();

System.out.print("Enter the Salary of Employee - ");

salary = (S)sc.next();

System.out.println();

}

public void displayEmp()

{

System.out.println("Employee Age -" + age + ", Salary - " + salary);

}

}

public class sample

{

public static void main(String[] args)

{

genricEmployee<Integer,Float> emp = new genricEmployee<Integer, Float>(100, 112220.0F);

emp.displayEmp();

genricEmployee<Integer,Integer> emp1 = new genricEmployee<Integer, Integer>();

emp1.acceptDisplay();

emp1.displayEmp();

}

}

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

Q:- WAP in java to perform below operation realated to hashmap.

1. Store mployee details by using employee class.

2. Traverse collection object

3. Delete employee details.

4. Update employee details.

Ans:-

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

//hashmap for removing duplicate keys

package javaapplication1;

import java.util.Scanner;

import java.util.\*;

/\*\*

 \*

 \* @author DUVISHAL

 \*/

class Employee1

{

    private int id;

    private String name;

    private int sal;

    private int age;

    public Employee1() {

    }

    public Employee1(int id, String name, int sal, int age) {

        this.id = id;

        this.name = name;

        this.sal = sal;

        this.age = age;

    }

    public void acceptDetails()

    {

        Scanner sc= new Scanner(System.in);

        System.out.print("Enter employee ID - ");

        id= sc.nextInt();

        System.out.println();

        System.out.print("Enter the Name of Employee - ");

        name = sc.next();

        System.out.println();

        System.out.print("Enter the Salary - ");

        sal=sc.nextInt();

        System.out.println();

        System.out.print("Enter the age of employee - )");

        age = sc.nextInt();

        System.out.println();

    }

    public void updateAge()

    {

        System.out.print("Enter the new age of the employee - ");

        this.age= new Scanner(System.in).nextInt();

         System.out.println();

    }

    public void displyDetails()

    {

        System.out.println("Employee ID - " + id + ",  Employee Name - " + name + " , Age - " + age + ", Salary - " + sal);

    }

    @Override

    public String toString()

    {

        return "Employee ID - " + id + ",  Employee Name - " + name + " , Age - " + age + ", Salary - " + sal;

    }

}

public class hashMap

{

    public static void main(String[] args)

    {

        int temp;

        HashMap<Integer, Employee1> hashmap = new HashMap<Integer, Employee1>();

        hashmap.put(100, new Employee1(100,"Vishal", 20000,25));

        hashmap.put(101, new Employee1(101,"Mehul", 27000, 24));

        hashmap.put(107, new Employee1(107,"Vivek", 21000, 30));

        hashmap.put( 109, new Employee1(109,"Shyam", 30000, 28));

        System.out.println("Hashmap contains below employee data.....");

         for(Map.Entry m:hashmap.entrySet())

         {

            System.out.print("Key - " + m.getKey()+ ", Value - " +m.getValue() );

            System.out.println();

         }

        System.out.print("Enter the employee id to remove - ");

        temp = new Scanner(System.in).nextInt();

        hashmap.remove(temp);

        System.out.println("Hashmap after removing emplpyee.....");

         for(Map.Entry m:hashmap.entrySet())

         {

            System.out.print("Key - " + m.getKey()+ ", Value - " +m.getValue() );

            System.out.println();

         }

         System.out.print("Enter the id of employee to change the age -");

         temp = new Scanner(System.in).nextInt();

         Employee1 tempEmp = hashmap.get(temp);

         tempEmp.updateAge();

         hashmap.put(temp, tempEmp);

         System.out.println("Hashmap after  emplpyee age update.....");

         for(Map.Entry m:hashmap.entrySet())

         {

            System.out.print("Key - " + m.getKey()+ ", Value - " +m.getValue() );

            System.out.println();

         }

    }

}

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

Q:- WAP in java to accept employee details bu using treeset. Enter the data in any manner but display the data

by arranging as per employeeID using camparator interface.

Ans:- /\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

//hashmap for removing duplicate keys

package javaapplication1;

import java.util.Scanner;

import java.util.\*;

/\*\*

 \*

 \* @author DUVISHAL

 \*/

class Employee1

{

    private int id;

    private String name;

    private int sal;

    private int age;

    public int getId() {

        return id;

    }

    public Employee1() {

    }

    public Employee1(int id, String name, int sal, int age) {

        this.id = id;

        this.name = name;

        this.sal = sal;

        this.age = age;

    }

    public void acceptDetails()

    {

        Scanner sc= new Scanner(System.in);

        System.out.print("Enter employee ID - ");

        id= sc.nextInt();

        System.out.println();

        System.out.print("Enter the Name of Employee - ");

        name = sc.next();

        System.out.println();

        System.out.print("Enter the Salary - ");

        sal=sc.nextInt();

        System.out.println();

        System.out.print("Enter the age of employee - )");

        age = sc.nextInt();

        System.out.println();

    }

    public void displyDetails()

    {

        System.out.println("Employee ID - " + id + ",  Employee Name - " + name + " , Age - " + age + ", Salary - " + sal);

    }

    @Override

    public String toString()

    {

        return "Employee ID - " + id + ",  Employee Name - " + name + " , Age - " + age + ", Salary - " + sal;

    }

}

class EmployeeIDComp implements Comparator<Employee1>

{

    @Override

    public int compare(Employee1 o, Employee1 s)

    {

        if(o.getId()==s.getId())

            return 0;

        else if(o.getId()> s.getId())

            return 1;

        else

            return -1;

    }

}

public class hashMap

{

    public static void main(String[] args)

    {

        TreeSet<Employee1> treesetEx = new TreeSet<Employee1>(new EmployeeIDComp());

        Employee1 emp1= new Employee1();

        Employee1 emp2= new Employee1();

        Employee1 emp3= new Employee1();

        Employee1 emp4= new Employee1();

        Employee1 emp5= new Employee1();

        emp1.acceptDetails();

        emp2.acceptDetails();

        emp3.acceptDetails();

        emp4.acceptDetails();

        emp5.acceptDetails();

        treesetEx.add(emp1);

        treesetEx.add(emp2);

        treesetEx.add(emp3);

        treesetEx.add(emp4);

        treesetEx.add(emp5);

        System.out.println("Tree set data in ascending order: ");

        Iterator iterator= treesetEx.iterator();

        while (iterator.hasNext())

        {

            System.out.println(iterator.next().toString() + "\n");

        }

    }

}

Q:- WAP to add employee details to file as an object.

Ans:-

package FileIOEX;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.List;

class Employee1 implements Serializable

{

    private double phone;

    private String name;

    private String address;

    private int age;

    public Employee1(double phone, String name, String address, int age) {

super();

this.phone = phone;

this.name = name;

this.address = address;

this.age = age;

}

    @Override

    public String toString()

    {

        return "Employee Name - " + name + ",  Employee Address - " + address + " , Employee Age - " + age + ", Employee Phone - " + phone;

    }

}

public class ObjectFileReadWrite

{

public static void main(String[] args)

{

FileOutputStream fobj= null;

ObjectOutputStream oobj= null;

FileInputStream fiobj= null;

ObjectInputStream oiobj= null;

try

{

fobj = new FileOutputStream("D:\\employeeObjFile.txt");

oobj = new ObjectOutputStream(fobj);

Employee1 obj1 = new Employee1(9999999999d, "Vishal", "Pune", 26);

Employee1 obj2 = new Employee1(8888888888d, "Sanket", "Nagpur", 25);

Employee1 obj3 = new Employee1(7777777777d, "Sanket", "Mumbai", 24);

oobj.writeObject(obj1);

oobj.writeObject(obj2);

oobj.writeObject(obj3);

}

catch (Exception e)

{

System.out.println("Exception occured - " + e.getMessage());

}

finally

{

if(oobj!= null)

try

{

oobj.close();

} catch (IOException e)

{

System.out.println("Exception occured - " + e.getMessage());

}

}

List<Object> objectList = new ArrayList<Object>();

try

{

fiobj = new FileInputStream("D:\\employeeObjFile.txt");

oiobj= new ObjectInputStream(fiobj);

while(true)

{

objectList.add(oiobj.readObject());

}

}

catch (Exception e)

{

System.out.println("Exception occured - " + e.getMessage());

}

finally

{

if(oiobj!= null)

try

{

oiobj.close();

} catch (IOException e)

{

System.out.println("Exception occured - " + e.getMessage());

}

}

for(Object emp:objectList)

{

System.out.println(emp.toString());

}

}

}

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

Q:- WAP in java to add StudentDetails to the file using DataStream.

Name - String, Age - int, Fee - float, Gender - Char

Ans:-

package FileIOEX;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.Serializable;

class Student

{

public String name;

public int age;

public float fee;

public char gender;

public Student(String name, int age, float fee, char gender) {

super();

this.name = name;

this.age = age;

this.fee = fee;

this.gender = gender;

}

public String getName() {

return name;

}

public int getAge() {

return age;

}

public float getFee() {

return fee;

}

public char getGender() {

return gender;

}

}

public class DataStreamFileReadWrite

{

public static void main(String[] args)

{

DataOutputStream doobj = null;

DataInputStream dioobj= null;

try

{

doobj = new DataOutputStream(new FileOutputStream("D:\\StudentObjDataStream.txt"));

Student st1 = new Student("Vishal",26,25000F,'M');

Student st2 = new Student("Sujata",25,21000F,'F');

doobj.writeUTF(st1.getName());

doobj.writeChar('\t');

doobj.writeInt(st1.getAge());

doobj.writeChar('\t');

doobj.writeFloat(st1.getFee());

doobj.writeChar('\t');

doobj.writeChar(st1.getGender());

doobj.writeChar('\t');

doobj.writeUTF(st2.getName());

doobj.writeChar('\t');

doobj.writeInt(st2.getAge());

doobj.writeChar('\t');

doobj.writeFloat(st2.getFee());

doobj.writeChar('\t');

doobj.writeChar(st2.getGender());

doobj.writeChar('\t');

}

catch(Exception e)

{

System.out.println("Exception occured - " + e.getMessage());

}

finally

{

if(doobj!= null)

try

{

doobj.close();

} catch (IOException e)

{

System.out.println("Exception occured - " + e.getMessage());

}

}

try

{

dioobj = new DataInputStream(new FileInputStream("D:\\StudentObjDataStream.txt"));

int studentcount=0;

while(dioobj.available() > 0)

{

String tempname = dioobj.readUTF();

dioobj.readChar();

int tempage = dioobj.readInt();

dioobj.readChar();

float tempfee = dioobj.readFloat();

dioobj.readChar();

char tempgender = dioobj.readChar();

dioobj.readChar();

System.out.println("Student " + (++studentcount) + " Name - " + tempname + ", Age - " + tempage + ", Fee - " + tempfee + ", Gender - "  + tempgender);

}

}

catch (Exception e)

{

System.out.println("Exception occured - " + e.getMessage());

}

finally

{

if(dioobj!= null)

try

{

dioobj.close();

} catch (IOException e)

{

System.out.println("Exception occured - " + e.getMessage());

}

}

}

}

Q:- Write a menu driven program in java to accept student details.

         perform below task.

          1. Store all data in file.

          2. Read the details only those student whose age is > 18 from the file and write it to another file.

Ans:-

package FileIOEX;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

class Student2 implements Serializable

{

    private int id;

    private String name;

    private int age;

    private double phone;

    private String feestatus;

    public Student2() {

    }

    public Student2(int id, String name, int age, double phone, String feestatus) {

        this.id = id;

        this.name = name;

        this.age = age;

        this.phone = phone;

        this.feestatus = feestatus;

    }

    public void acceptDetails()

    {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter Student ID - ");

        this.id = sc.nextInt();

        System.out.println("");

        System.out.print("Enter Student Name - ");

        this.name = sc.next();

        System.out.println("");

        System.out.print("Enter Student Age - ");

        this.age = sc.nextInt();

        System.out.println("");

        System.out.print("Enter Student Phone - ");

        this.phone = sc.nextDouble();

        System.out.println("");

        System.out.print("Enter FeeStatus - ");

        this.feestatus = sc.next();

        System.out.println("");

    }

    public int getAge() {

        return age;

    }

    @Override

    public String toString()

    {

    return "Student Details -> Name = " + name + ", ID = " + id + ", Age =  " + age + ", Phone = " + phone + ", FeeStatus = " + feestatus;

    }

}

public class StudentReadWrite

{

  public static FileOutputStream fobj= null;

   public static ObjectOutputStream oobj= null;

   public static FileInputStream fiobj= null;

   public static ObjectInputStream oiobj= null;

   public static List<Object> objectList;

   public static void main(String[] args)

   {

       try

       {

           fobj = new FileOutputStream("D:\\studentDetails.txt");

           oobj = new ObjectOutputStream(fobj);

           fiobj = new FileInputStream("D:\\studentDetails.txt");

           oiobj= new ObjectInputStream(fiobj);

           objectList = new ArrayList<Object>();

           while(true)

           {

               System.out.println("1. Add Student Details.");

               System.out.println("2. Read Student Details.");

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

               System.out.print("Ënter your choice = ");

               int choice = (new Scanner(System.in)).nextInt();

               switch(choice)

               {

                  case 1:  addStudent();

                           break;

                  case 2:  readStudent();

                  readList();

                           break;

                  case 3:  exitFunction();

                  System.exit(0);

                           break;

                  default:System.out.println("Wrong option, please try again...");

                          break;

               }

           }

       }

       catch (Exception e)

       {

           System.out.println("Exception occured1 - " + e.getMessage());

       }

   }

   public static void addStudent()

   {

       try

       {

        readStudent();

        Student2 temp = new Student2();

        temp.acceptDetails();

        objectList.add(temp);

           for(Object std:objectList)

            oobj.writeObject(std);

       }

       catch (Exception e)

       {

           System.out.println("Exception occured2 - " + e.getMessage());

       }

       objectList.clear();

   }

   public static void readStudent()

   {

       try

       {

while(true)

{

objectList.add(oiobj.readObject());

}

}

       catch (Exception e)

       {

        System.out.println("Exception occured3 - " + e.getMessage());

}

   }

   public static void exitFunction()

   {

    try

    {

    if(oobj!= null)

    oobj.close();

    if(oiobj!= null)

    oiobj.close();

    }

    catch (Exception e)

    {

    System.out.println("Exception occured4 - " + e.getMessage());

    }

    objectList.clear();

    objectList=null;

}

   public static void readList()

   {

    FileOutputStream fobj1;

    ObjectOutputStream oobj1;

    try

    {

    fobj1= new FileOutputStream("D:\\studentDetails1.txt");;

    oobj1= new ObjectOutputStream(fobj1);

   for(Object std:objectList)

{

    if(((Student2)std).getAge() > 18)

    {

    System.out.println(std.toString());

    oobj1.writeObject(std);

    }

}

   try

    {

    if(oobj1!= null)

    oobj1.close();

    }

    catch (Exception e)

    {

    System.out.println("Exception occured4 - " + e.getMessage());

    }

    }

    catch (Exception e)

    {

    System.out.println("Exception occured4 - " + e.getMessage());

    }

   }

}

Q:- Write a program in java to accept customer details and perform below task.

       1. Store the all data in file customer.txt

       2.Read the those details those customer whose salary is > 100000 and write it to another file.

Ans:-

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

package javaapplication1;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

/\*\*

 \*

 \* @author DUVISHAL

 \*/

class customer implements Serializable

{

    private int customerID;

    private String customerName;

    private int customerAge;

    private long customerPhone;

    private long customerSalary;

    private String customerCity;

    private String customerCountry;

    public void acceptDetails()

    {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the Customer ID - ");

        customerID = sc.nextInt();

        System.out.println();

        System.out.print("Enter the Customer Name - ");

        customerName = sc.next();

        System.out.println();

        System.out.print("Enter the Customer Age - ");

        customerAge = sc.nextInt();

        System.out.println();

        System.out.print("Enter the Customer Phone - ");

        customerPhone = sc.nextLong();

        System.out.println();

        System.out.print("Enter the Customer Salary - ");

        customerSalary = sc.nextLong();

        System.out.println();

        System.out.print("Enter the Customer City - ");

        customerCity = sc.next();

        System.out.println();

        System.out.print("Enter the Customer Country - ");

        customerCountry = sc.next();

        System.out.println();

    }

    @Override

    public String toString() {

        return "customer{" + "customerID=" + customerID + ", customerName=" + customerName + ", customerAge=" + customerAge + ", customerPhone=" + customerPhone + ", customerSalary=" + customerSalary + ", customerCity=" + customerCity + ", customerCountry=" + customerCountry + '}';

    }

    public long getCustomerSalary() {

        return customerSalary;

    }

}

public class CustomerFileIO

{

   public static FileOutputStream fobj= null;

   public static ObjectOutputStream oobj= null;

   public static FileInputStream fiobj= null;

   public static ObjectInputStream oiobj= null;

   public static List<Object> objectList;

   public static void main(String[] args)

   {

       try

       {

           fobj = new FileOutputStream("D:\\CustomerDetails.txt");

           oobj = new ObjectOutputStream(fobj);

           fiobj = new FileInputStream("D:\\CustomerDetails.txt");

           oiobj= new ObjectInputStream(fiobj);

           objectList = new ArrayList<Object>();

           while(true)

           {

               System.out.println("1. Add Customer  Details.");

               System.out.println("2. Read Customer Details.");

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

               System.out.print("Ënter your choice = ");

               int choice = (new Scanner(System.in)).nextInt();

               switch(choice)

               {

                  case 1:  addCustomer();

                           break;

                  case 2:  readCustomer();

                           readList();

                           break;

                  case 3:  exitFunction();

                  System.exit(0);

                           break;

                  default:System.out.println("Wrong option, please try again...");

                          break;

               }

           }

       }

       catch (Exception e)

       {

           System.out.println("Exception occured1 - " + e.getMessage());

       }

   }

   public static void addCustomer()

   {

       try

       {

        readCustomer();

        customer temp = new customer();

        temp.acceptDetails();

        objectList.add(temp);

           for(Object std:objectList)

            oobj.writeObject(std);

       }

       catch (Exception e)

       {

           System.out.println("Exception occured2 - " + e.getMessage());

       }

       objectList.clear();

   }

   public static void readCustomer()

    {

        try

        {

            while(true)

            {

                objectList.add(oiobj.readObject());

            }

        }

        catch (Exception e)

        {

         System.out.println("Exception occured3 - " + e.getMessage());

        }

   }

   public static void exitFunction()

   {

    try

    {

        if(oobj!= null)

        oobj.close();

        if(oiobj!= null)

        oiobj.close();

    }

    catch (Exception e)

    {

        System.out.println("Exception occured4 - " + e.getMessage());

    }

    objectList.clear();

    objectList=null;

}

   public static void readList()

   {

        FileOutputStream fobj1;

        ObjectOutputStream oobj1;

        try

        {

            fobj1= new FileOutputStream("D:\\Customer.txt");;

            oobj1= new ObjectOutputStream(fobj1);

            for(Object cst:objectList)

            {

                if(((customer)cst).getCustomerSalary()> 100000)

                {

                System.out.println(cst.toString());

                oobj1.writeObject(cst);

                }

            }

            try

             {

             if(oobj1!= null)

             oobj1.close();

             }

             catch (Exception e)

             {

             System.out.println("Exception occured4 - " + e.getMessage());

             }

        }

        catch (Exception e)

        {

            System.out.println("Exception occured4 - " + e.getMessage());

        }

   }

}