**Java Basic Assignment Questions**

**Q1.**Write a Java program to return true if a given string contain the string 'pop', but the middle 'o' also may other character.

**SOLUTION**

**public class Main {**

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

**{**

**String s="spaps";**

**boolean flag=false;**

**for(char i='a';i<='z';i++)**

**{**

**String seq="p"+i+"p";**

**if(s.contains(seq))**

**{**

**flag = true;**

**break;**

**}**

**}**

**if(flag)**

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

**else**

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

**}**

**}**

**Q2.**Write a Java program to sort a given binary array in linear times.

Linear time: An algorithm is said to take linear time, or O(n) time, if its time complexity is O(n). Informally, this means that the running time increases at most linearly with the size of the input. More precisely, this means that there is a constant c such that the running time is at most cn for every input of size n. For example, a procedure that adds up all elements of a list requires time proportional to the length of the list, if the adding time is constant, or, at least, bounded by a constant.

Example:

Input :

b\_nums[] = { 0, 1, 1, 0, 1, 1, 0, 1, 0, 0 }

Output:

After sorting: [0, 0, 0, 0, 0, 1, 1, 1, 1, 1]  
  
  
**SOLUTION**  
**public class Main {**

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

**int b\_nums[] = { 0, 1, 1, 0, 1, 1, 0, 1, 0, 0 };**

**int count0=0,count1=1;**

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

**{**

**if(b\_nums[i]==0)**

**count0++;**

**else if(b\_nums[i]==1)**

**count1++;**

**}**

**for(int i=0;i<count0;i++)**

**b\_nums[i]=0;**

**for(int i=count0;i<b\_nums.length;i++)**

**b\_nums[i]=1;**

**for(int i:b\_nums)**

**System.out.print(i+" ");**

**}**

**}**

**Q3.** Write a Java program to remove a specified character from a given string.

Sample Output:

Original string: abcdefabcdeabcdaaa

Second string: bcdefbcdebcd  
  
**SOLUTION**

**public class Third {**

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

**String s="abcdefabcdeabcdaaa";**

**char toBeRemoved='a';**

**StringBuilder sb=new StringBuilder("");**

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

**{**

**if(s.charAt(i)!=toBeRemoved)**

**sb.append(s.charAt(i));**

**}**

**s=new String(sb);**

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

**}**

**}**

**Q4.**Write a Java Program to reverse a string without using String inbuilt function.  
  
**SOLUTION  
public class Main {**

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

**String s="Sample String";**

**char[] arr=s.toCharArray();**

**int l=0,r=arr.length-1;**

**while(l<r)**

**{**

**char temp=arr[l];**

**arr[l]=arr[r];**

**arr[r]=temp;**

**l++;**

**r--;**

**}**

**s=new String(arr);**

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

**}**

**}**

**Q5.**First Factorial

Have the function FirstFactorial(num) take the num parameter being passed and return the factorial of it. For example: if num = 4, then your program should return (4 \* 3 \* 2 \* 1) = 24. For the test cases, the range will be between 1 and 18 and the input will always be an integer.

#### Examples

Input: 4

Output: 24

Input: 8

Output: 40320  
  
**SOLUTION**

**public class Main {**

**static int firstFactorial(int num)**

**{**

**if(num<=1)**

**return 1;**

**return num\*firstFactorial(num-1);**

**}**

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

**int factorial=firstFactorial(8);**

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

**}**

**}**

**Q6.** Write a program to find the number of occurrences of the duplicate words in a string and print them?

**SOLUTION**

**import java.util.HashMap;**

**import java.util.Map;**

**public class Main {**

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

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

**String s="This is duplicate check sentence. This is duplicate check sentence.";**

**String[] arr=s.split("\\s");**

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

**{**

**if(!map.containsKey(arr[i]))**

**map.put(arr[i],1);**

**else**

**map.put(arr[i],map.get(arr[i])+1);**

**}**

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

**}**

**}**

**Q7.** Write a program to find the number of occurrences of a character in a string without using a loop?

String.  
**SOLUTION  
package Ques7;**

**import java.util.HashMap;**

**import java.util.Map;**

**public class Main {**

**static void checkOccurrences(String s, int index,Map<Character,Integer> map)**

**{**

**if(index>=s.length())**

**return;**

**if(!map.containsKey(s.charAt(index)))**

**map.put(s.charAt(index),1);**

**else**

**map.put(s.charAt(index),map.get(s.charAt(index))+1);**

**checkOccurrences(s,index+1,map);**

**}**

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

**Map<Character,Integer> map=new HashMap<Character,Integer>();**

**String s="abcdefghijabcde";**

**checkOccurrences(s,0,map);**

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

**}**

**}**

**Q8.**Write a program to print your Firstname, LastName & age using static block, static method & static variable respectively.  
**SOLUTION  
package Ques8;**

**public class Main {**

**//STATIC VARIABLES**

**static String firstName="Prakhar";**

**static String lastName="Rastogi";**

**static int age=24;**

**static**

**{**

**//USING STATIC BLOCK**

**System.out.println("PRINT 1 USING STATIC BLOCK");**

**System.out.println("First Name:"+firstName+" , Last Name:"+lastName+" , Age:"+age);**

**}**

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

**//USING STATIC METHOD MAIN**

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

**System.out.println("PRINT 2 USING STATIC MAIN METHOD");**

**System.out.println("First Name:"+firstName+" , Last Name:"+lastName+" , Age:"+age);**

**}**

**}**

**Q9.**Create 3 subclasses of bank SBI, BOI, ICICI all 4 should have a method called get details which provide there specific details like rate of interest etc, print details of every bank.  
**SOLUTION  
  
//ABSTRACT CLASS BANK   
  
package Ques9;**

**public abstract class Bank {**

**abstract public void getDetails();**

**}**

**//CLASS BOI EXTENDS BANK**  
**package Ques9;**

**public class BOI extends Bank{**

**String name="BOI";**

**int rateOfInterest=7;**

**@Override**

**public void getDetails() {**

**System.out.println("Name:"+name+", Rate Of Interest:"+rateOfInterest);**

**}**

**}**

**//CLASS ICICI EXTENDS BANK  
package Ques9;**

**public class ICICI extends Bank{**

**String name="ICICI";**

**int rateOfInterest=8;**

**@Override**

**public void getDetails() {**

**System.out.println("Name:"+name+", Rate Of Interest:"+rateOfInterest);**

**}**

**}**

**//CLASS SBI EXTENDS BANK**

**package Ques9;**

**public class SBI extends Bank {**

**String name="SBI";**

**int rateOfInterest=6;**

**@Override**

**public void getDetails() {**

**System.out.println("Name:"+name+", Rate Of Interest:"+rateOfInterest);**

**}**

**}**

**//MAIN CLASS  
package Ques9;**

**public class Main {**

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

**Bank bank=new BOI();**

**bank.getDetails();**

**bank=new ICICI();**

**bank.getDetails();**

**bank=new SBI();**

**bank.getDetails();**

**}**

**}  
  
Q10.** WAP to read words from the keyboard until the word done is entered. For each word except done, report whether its first character is equal to its last character. For the required loop, use a

1. while statement
2. do-while statement

**SOLUTION  
package Ques10;**

**import java.util.Scanner;**

**public class Main {**

**static void testWhileLoop()**

**{**

**System.out.println("CHECKING WHILE LOOP FUNCTIONALITY");**

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

**String s=sc.next();**

**while(!s.equalsIgnoreCase("done"))**

**{**

**if(s.charAt(0)==s.charAt(s.length()-1))**

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

**else**

**System.out.println("Not Equal");**

**s=sc.next();**

**}**

**}**

**static void testDoWhileLoop()**

**{**

**System.out.println("CHECKING DO-WHILE LOOP FUNCTIONALITY");**

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

**String s=sc.next();**

**if(s.equalsIgnoreCase("done"))**

**return;**

**do{**

**if(s.charAt(0)==s.charAt(s.length()-1))**

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

**else**

**System.out.println("Not Equal");**

**s=sc.next();**

**}**

**while(!s.equalsIgnoreCase("done"));**

**}**

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

**testWhileLoop();**

**testDoWhileLoop();**

**}**

**}**

**Q11.** Design classes having attributes and methods (only skeleton) for a coffee shop. There are three different actors in our scenario and I have listed the different actions they do also below

\* Customer

- Pays the cash to the cashier and places his order, get a token number back

- Waits for the intimation that order for his token is ready

- Upon intimation/notification he collects the coffee and enjoys his drink

( Assumption: Customer waits till the coffee is done, he won’t timeout and cancel the order. The customer always likes the drink served. Exceptions like he not liking his coffee, he got wrong coffee are not considered to keep the design simeep the design simple.)

\* Cashier

- Takes order and payment from the customer

- Upon payment, creates an ordexr and places it into the order queue

- Intimates the customer that he has to wait for his token and gives him his token

( Assumption: Token returned to the customer is the order id. Order queue is unlimited. With a simple modification, we can design for a limited queue size)

\* Barista

- Gets the next order from the queue

- Prepares the coffee

- Places the coffee in the cozmpleted order queue

- Places a notification that order for token is ready

**CODE SKELETON  
  
package Ques11;**

**public class Customer {**

**/\***

**- Pays the cash to the cashier and places his order, get a token number back**

**- Waits for the intimation that order for his token is ready**

**- Upon intimation/notification he collects the coffee and enjoys his drink**

**\*/**

**private String tokenNumber;**

**private boolean isReadyIntimation;**

**public void pay()**

**{**

**//pay the cash to the cashier**

**//cashier initializes the tokenNumber variable of customer**

**}**

**public void collectOrder()**

**{**

**//collect order / coffee**

**}**

**}**

**package Ques11;**

**import java.util.ArrayDeque;**

**import java.util.List;**

**public class Cashier {**

**/\***

**- Takes order and payment from the customer**

**- Upon payment, creates an order and places it into the order queue**

**- Intimates the customer that he has to wait for his token and gives him his token**

**\*/**

**//Association - has-a relation**

**List<Customer> customerList;**

**public void recieveOrderAndPayment(){}**

**}**

**package Ques11;**

**import java.util.ArrayDeque;**

**public class Barista {**

**/\***

**- Gets the next order from the queue**

**- Prepares the coffee**

**- Places the coffee in the completed order queue**

**- Places a notification that order for token is ready**

**\*/**

**ArrayDeque<String> orders=new ArrayDeque<>();**

**public void prepareCoffee()**

**{**

**String currentOrder= orders.getFirst();**

**//prepare order**

**}**

**public void notifyCustomerForOrderCompletion()**

**{**

**}**

**}**

**Q12.** Create a custom exception that do not have any stack trace  
**SOLUTION  
package Ques12;**

**class CustomException extends Exception**

**{**

**CustomException(String msg)**

**{**

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

**}**

**}**

**public class Main {**

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

**try {**

**exceptionFunc();**

**} catch(Throwable e) {**

**StackTraceElement[] trace = e.getStackTrace();**

**System.err.println(trace[0].toString());**

**}**

**}**

**public static void exceptionFunc()throws CustomException {**

**CustomException e=new CustomException("Throwing Exception without stack trace");**

**StackTraceElement[] trace = new StackTraceElement[] {**

**new StackTraceElement("ClassName","methodName","fileName",1)**

**};**

**// sets the stack trace elements**

**e.setStackTrace(trace);**

**throw e;**

**}**

**}**

**Q13.** Create Java classes having suitable attributes for Library management system.Use OOPs concepts in your design.Also try to use interfaces and abstract classes.

**SOLUTION  
  
package Ques13;**

**import java.util.Scanner;**

**class Library {**

**static String bookname, issuedate, date;**

**static int bookid, total\_issuebook;**

**void add() {**

**System.out.println("Enter the book name, Prize and Book\_No");**

**Scanner obj2 = new Scanner(System.in);**

**String str = obj2.next();**

**float price = Float.parseFloat(obj2.next());**

**int bookno = Integer.parseInt(obj2.next());**

**System.out.println("your details are Name: " + str + " Price: " + price + " Book\_No: " + bookno);**

**}**

**void iss() {**

**Scanner obj3 = new Scanner(System.in);**

**System.out.println("Book Name");**

**bookname = obj3.nextLine();**

**System.out.println("Book\_id");**

**bookid = obj3.nextInt();**

**obj3.nextLine();**

**System.out.println("issue date");**

**issuedate = obj3.nextLine();**

**System.out.println("total no of book issued");**

**total\_issuebook = obj3.nextInt();**

**obj3.nextLine();**

**System.out.println("Return book date");**

**String date = obj3.nextLine();**

**}**

**int getid() {**

**return bookid;**

**}**

**void ret() {**

**System.out.println("Enter the Student\_Name and book\_id");**

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

**String name = c.nextLine();**

**int bookId = c.nextInt();**

**if (bookid == bookId) {**

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

**System.out.println("Book\_name:" + Library.bookname);**

**System.out.println("Book\_id:" + Library.bookid);**

**System.out.println("Issue date:" + Library.issuedate);**

**System.out.println("Total number of book issued:" + Library.total\_issuebook);**

**System.out.println("Book return date" + Library.date);**

**} else {**

**System.out.println("You entered the wrong id , Please dd the correct id kindly.");**

**}**

**}**

**void details() {**

**System.out.println("Book\_name:" + Library.bookname);**

**System.out.println("Book\_id:" + Library.bookid);**

**System.out.println("Issue date:" + Library.issuedate);**

**System.out.println("Total number of book issued:" + Library.total\_issuebook);**

**System.out.println("Book return date" + Library.date);**

**}**

**void exit() {**

**}**

**}**

**package Ques13;**

**import java.util.Scanner;**

**class LibraryManagementSystem {**

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

**char r;**

**do {**

**System.out.println("Library Management System\*");**

**System.out.println("Press 1 to add book");**

**System.out.println("Press 2 to issue a book");**

**System.out.println("Press 3 to return a book");**

**System.out.println("Press 4 to print complete issue details");**

**System.out.println("Press 5 to exit");**

**Scanner obj1 = new Scanner(System.in);**

**System.out.println("Enter any Number");**

**int option = obj1.nextInt();**

**switch (option) {**

**case 1:**

**Library addbook\_obj = new Library();**

**addbook\_obj.add();**

**break;**

**case 2:**

**Library issuebook\_obj = new Library();**

**issuebook\_obj.iss();**

**break;**

**case 3:**

**Library returnbook\_obj = new Library();**

**returnbook\_obj.ret();**

**break;**

**case 4:**

**Library detail\_obj = new Library();**

**detail\_obj.details();**

**break;**

**case 5:**

**Library add = new Library();**

**add.exit();**

**break;**

**default:**

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

**}**

**System.out.println("Do you want to select next option(y/n)");**

**r = obj1.next().charAt(0);**

**}while (r == 'Y' || r == 'y') ;**

**if (r == 'N' || r == 'n') {**

**Library z = new Library();**

**z.exit();**

**}**

**}**

**}**

**Q14.**WAP to produce NoClassDefFoundError and ClassNotFoundException exception.

**SOLUTION  
package Ques13;**

**public class Main {**

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

**try**

**{**

**Class.forName("ClassNotFoundException");**

**}**

**catch (ClassNotFoundException ex)**

**{**

**ex.printStackTrace();**

**}**

**}**

**}**

**Q15.** WAP to create a singleton class.

**SOLUTION  
package Ques15;**

**import com.sun.xml.internal.bind.v2.runtime.unmarshaller.XsiNilLoader;**

**class Singleton**

**{**

**private static Singleton singleton\_instance=null;**

**private Singleton()**

**{**

**System.out.println("Singleton constructor");**

**}**

**public static Singleton getInstance()**

**{**

**if(singleton\_instance==null)**

**singleton\_instance=new Singleton();**

**return singleton\_instance;**

**}**

**}**

**public class Main**

**{**

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

**Singleton x= Singleton.getInstance();**

**System.out.println("Singleton Instance created!!!");**

**}**

**}**

**Q16.** WAP to show object cloning in java using cloneable and copy constructor both.

**SOLUTION  
package Ques16;**

**class A implements Cloneable**

**{**

**int i;**

**String s;**

**//CLONEABLE INTERFACE**

**A(int i,String s)**

**{**

**this.i=i;**

**this.s=s;**

**}**

**//COPY CONSTRUCTOR**

**A(A obj)**

**{**

**this.i=obj.i;**

**this.s=obj.s;**

**}**

**@Override**

**protected Object clone() throws CloneNotSupportedException {**

**return super.clone();**

**}**

**}**

**public class Main {**

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

**A a1=new A(1,"A");**

**System.out.println("a1.i: "+a1.i+" ,a1.s: "+a1.s);**

**A a2= (A) a1.clone();**

**System.out.println("<----Clone interface implementation----> ");**

**System.out.println("a2.i: "+a2.i+" ,a2.s: "+a2.s);**

**A a3=new A(a2);**

**System.out.println("<----Copy constructor implementation---->");**

**System.out.println("a3.i: "+a3.i+" ,a3.s: "+a3.s);**

**}**

**}**

**Q17.** WAP showing try, multi-catch and finally blocks.  
**SOLUTION  
package Ques17;**

**import java.util.InputMismatchException;**

**public class Main {**

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

**boolean flag=false;**

**try**

**{**

**int x=1000/0;**

**}**

**catch(ArrayIndexOutOfBoundsException e)**

**{**

**flag=true;**

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

**e.printStackTrace();**

**}**

**catch(InputMismatchException e)**

**{**

**flag=true;**

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

**e.printStackTrace();**

**}**

**catch(ArithmeticException e)**

**{**

**flag=true;**

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

**e.printStackTrace();**

**}**

**catch(Exception e)**

**{**

**flag=true;**

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

**e.printStackTrace();**

**}**

**finally**

**{**

**if(flag)**

**System.out.println("Exception catched");**

**else**

**System.out.println("No Exception catched");**

**}**

**}**

**}**

**Q18.**Using (instance) Method reference create and apply add and subtract method and using (Static) Method reference create and apply multiplication method for the functional interface created.

Create an Employee Class with instance variables (String) name, (Integer)age, (String)city and get the instance of the Class using constructor reference

Implement following functional interfaces from java.util.function using lambdas:

(1) Consumer

(2) Supplier

(3) Predicate

(4) Function

-Create and access default and static method of an interface.

-Override the default method of the interface.

-Implement multiple inheritance with default method inside interface.  
  
  
**SOLUTION  
  
package Ques18;**

**import java.util.ArrayList;**

**import java.util.Arrays;**

**import java.util.List;**

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

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

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

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

**import java.util.stream.Collectors;**

**import java.util.stream.Stream;**

**public class Main {**

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

**EmployeeFactory empFactory = Employee::new;**

**Employee emp1 = empFactory.getEmployee("Prakhar", 25, "Delhi");**

**Employee emp2 = empFactory.getEmployee("Rastogi", 25, "New Delhi");**

**// Printer.print(emp1);**

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

**employees.add(emp1.toString());**

**employees.add(emp2.toString());**

**// ===================== Using Consumer interface =============================**

**// Referring method to String type Consumer interface**

**Consumer<String> printTxt = ConsumerInterface::printMessage;**

**printTxt.accept("Hello from consumer"); // Calling Consumer method**

**// printTxt.accept(employees.toString());**

**Consumer<List<String>> employeePrint = (t) -> System.out.println(t);**

**Stream<List<String>> empdetail = Stream.of((employees));**

**empdetail.forEach(employeePrint);**

**// By using default andThen method of Consumer interface**

**Consumer<Integer> display = a -> System.out.println(a);**

**// Consumer to multiply 2 to every integer of a list**

**Consumer<List<Integer> > modify = list ->**

**{**

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

**list.set(i, 2 \* list.get(i));**

**};**

**// Consumer to display a list of integers**

**Consumer<List<Integer> >**

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

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

**list.add(2);**

**list.add(1);**

**list.add(3);**

**// using addThen()**

**modify.andThen(dispList).accept(list);**

**// =================== Supplier Interface ========================================================**

**// This function returns a random value.**

**Supplier<Boolean> empty = () -> employees.isEmpty();**

**// Print the random value using get()**

**printTxt.accept("\nUsing get() method of Supplier Interface : Is employee list empty : "+empty.get());**

**// ======================= Using Function Interface ===============================================**

**Function<String , Integer> charOfList = a -> a.length();**

**// apply the function to get the result**

**List<Integer> list1 = employees.stream().map(charOfList).collect(Collectors.toList());**

**printTxt.accept("Total characters in list:"+charOfList.apply(employees.toString()));**

**// ============================== Using Predicate Interface =====================================**

**Predicate<String> startPredicate = str -> str.startsWith("A");**

**Predicate<String> lengthPredicate = str -> str.length() >= 10;**

**employees.stream().filter(startPredicate.and(lengthPredicate)).forEach(System.out::println);**

**// \*\*\*\*\*\*\*Calling Predicate test() method \*\*\*\*\*\*\*\*\*\*\*\*\***

**// Creating predicate**

**Predicate<Integer> lesserthan = age -> (age < 18);**

**printTxt.accept("Using Predicate test() method : "+lesserthan.test(18)+"");**

**// \*\*\*\*\*\*\*\* Calling Predicate and() method \*\*\*\*\*\*\*\*\*\*\*\***

**Predicate<Integer> noGreaterThan5 = x -> x > 5;**

**Predicate<Integer> noLessThan8 = x -> x < 8;**

**List<Integer> listArr = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);**

**List<Integer> collect = listArr.stream()**

**.filter(noGreaterThan5.and(noLessThan8))**

**.collect(Collectors.toList());**

**printTxt.accept("Using Predicate default and() method: "+collect);**

**}**

**}**

**package Ques18;**

**public class Employee {**

**String name;**

**String city;**

**int age;**

**Employee (String name, int age, String city){**

**this.name = name;**

**this.age = age;**

**this.city = city;**

**}**

**@Override**

**public String toString() {**

**return "Name = " + name +**

**", Age = " + age +**

**", City = " + city;**

**}**

**}**

**package Ques18;**

**@FunctionalInterface**

**public interface EmployeeFactory {**

**public abstract Employee getEmployee(String name, Integer age, String city);**

**}**

**package Ques18;**

**public interface ConsumerInterface {**

**static void printMessage(String text){**

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

**}**

**}**

**Q19.**Convert a non-negative integer num to its English words representation.

Input: num = 123

Output: "One Hundred Twenty Three"

Input: num = 1234567891

Output: "One Billion Two Hundred Thirty Four Million Five Hundred Sixty

Seven Thousand Eight Hundred Ninety One"

**SOLUTION  
package Ques19;**

**class Solution**

**{**

**private static String[] lessThanTwenty = new String[] {"", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen"};**

**private static String[] tens = new String[]{"", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"};**

**public String numberToWords(int num) {**

**if(num == 0) {**

**return "Zero";**

**}**

**return helper(num);**

**}**

**private String helper(int num) {**

**StringBuilder res = new StringBuilder();**

**if(num >= 1000000000) {**

**res.append(helper(num/1000000000)).append(" Billion ").append(helper(num%1000000000));**

**} else if(num >= 1000000) {**

**res.append(helper(num/1000000)).append(" Million ").append(helper(num%1000000));**

**} else if(num >= 1000) {**

**res.append(helper(num/1000)).append(" Thousand ").append(helper(num%1000));**

**} else if(num >= 100) {**

**res.append(helper(num/100)).append(" Hundred ").append(helper(num%100));**

**} else if(num >=20) {**

**res.append(tens[num/10-1]).append(" ").append(helper(num%10));**

**} else {**

**res.append(lessThanTwenty[num]);**

**}**

**return res.toString().trim();**

**}**

**}**

**public class Main {**

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

**Solution solution=new Solution();**

**String result=solution.numberToWords(1234567891);**

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

**}**

**}**

**Q20.**This assignment is supposed to make you practice your skills on interfaces, classes and OOPs.

Story - The system is a duck simulation game. There are ducks, each having weight and number of wings. All ducks can fly and quack, but different type of ducks fly and quack differently. For instance, let us consider the following–

1.Rubber ducks don’t fly and squeak.

2.Mallard ducks fly fast and quack loud.

3.Redhead ducks fly slow and quack mild.

All ducks have following common property:

Type of Duck – Use enum to represent this value

Design classes and interfaces for the above story to meet the following requirements -

a.Create a duck

b.Show details of a duck, i.e. when you call for e.g. ShowDetails() method of a duck, duck should print its traits.

User following concepts: class, interfaces, polymorphism etc.

**SOLUTION  
  
  
  
  
  
package Ques20;**

**public class Bird {**

**enum DuckType{**

**Rubber,**

**Mallard,**

**RedHead;**

**}**

**}**

**package Ques20;**

**public interface BirdInterface {**

**void ShowDetail();**

**}**

**package Ques20;**

**public class Duck implements BirdInterface{**

**private double weight;**

**private int numberOfWings;**

**private Bird.DuckType TypeOfDuck;**

**public Duck (double weight, int numberOfWings, Bird.DuckType TypeOfDuck){**

**this.weight = weight;**

**this.numberOfWings = numberOfWings;**

**this.TypeOfDuck = TypeOfDuck;**

**}**

**@Override**

**public void ShowDetail() {**

**if ( TypeOfDuck == Bird.DuckType.Mallard)**

**System.out.println("Mallard Duck");**

**else if ( TypeOfDuck == Bird.DuckType.Rubber)**

**System.out.println("Rubber Duck");**

**else if ( TypeOfDuck == Bird.DuckType.RedHead)**

**System.out.println("Redhead Duck");**

**System.out.println("Weight of duck : " + weight);**

**System.out.println("Total number of wings : " + numberOfWings);**

**}**

**}**

**package Ques20;**

**public class MallardDuck extends Duck{**

**public MallardDuck(double weight, int numberOfWings, Bird.DuckType TypeOfDuck) {**

**super(weight, numberOfWings, TypeOfDuck);**

**}**

**public void ShowDetail(){**

**super.ShowDetail();**

**System.out.println("Mallard ducks fly fast and quack loud.");**

**}**

**}**

**package Ques20;**

**public class RedheadDuck extends Duck{**

**public RedheadDuck(double weight, int numberOfWings, Bird.DuckType TypeOfDuck) {**

**super(weight, numberOfWings, TypeOfDuck);**

**}**

**public void ShowDetail(){**

**super.ShowDetail();**

**System.out.println("Redhead ducks fly slow and quack mild.");**

**}**

**}**

**package Ques20;**

**public class RubberDuck extends Duck{**

**public RubberDuck(double weight, int numberOfWings, Bird.DuckType TypeOfDuck) {**

**super(weight, numberOfWings, TypeOfDuck);**

**}**

**public void ShowDetail(){**

**super.ShowDetail();**

**System.out.println("Rubber ducks don’t fly and squeak.");**

**}**

**}**

**package Ques20;**

**public class Main {**

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

**BirdInterface[] duck = new BirdInterface[3];**

**duck[0] = new RubberDuck(2,4, Bird.DuckType.Rubber);**

**duck[1] = new MallardDuck(4,6, Bird.DuckType.Mallard);**

**duck[2] = new RedheadDuck(6,8, Bird.DuckType.RedHead);**

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

**duck[i].ShowDetail();**

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

**}**

**}**

**}**

**Q21.**Write a java program to maintain the list of Employees who works in the organization. Follow the below given design to develop the above program:

Design a class Employee with EmpId, EmpName & EmpDesignation as private

attributes. It contains a parameterized constructor to initialize all the data members of class. It also comprises the Getter methods for all the private fields.

Design a main class comprising main method. The main class comprises an array of Employee object which needs to be initialized with the command line input.

It displays the information of Employees from the array using loop.

**SOLUTION  
package Ques21;**

**class Employee**

**{**

**private long empId;**

**private String empName;**

**private String designation;**

**public Employee(long empId, String empName, String designation) {**

**this.empId = empId;**

**this.empName = empName;**

**this.designation = designation;**

**}**

**public long getEmpId() {**

**return empId;**

**}**

**public String getEmpName() {**

**return empName;**

**}**

**public String getDesignation() {**

**return designation;**

**}**

**}**

**public class Main {**

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

**if(args.length%3!=0)**

**{**

**System.out.println("Please check number of arguments");**

**return;**

**}**

**Employee[] emp=new Employee[args.length/3];**

**int index=0;**

**try {**

**for (int i = 0; i < args.length; i += 3) {**

**Employee temp = new Employee(Long.valueOf(args[i]), args[i + 1], args[i + 2]);**

**emp[index] = temp;**

**index++;**

**}**

**}**

**catch(Exception e)**

**{**

**e.printStackTrace();**

**}**

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

**{**

**System.out.println("ID: "+emp[i].getEmpId()+" , Name: "+emp[i].getEmpName()+" , Designation: "+emp[i].getDesignation());**

**}**

**}**

**}**

**Q22.**Check Digits. Credit cards usually have a so-called check digit. This is a single digit that I assigned when the account number is developed and has a special property. One particularly simple mechanism is to assign the last digit of the sum of all the other digits. For example, suppose we have a nine-digit account number (including the check digit). The check digit would be the sum of the eight digits. This digit could be placed any where’s in the sequence, say the

third digit. As a full example, suppose the eight numbers are 12345678. Their sum is 36; thus, 6 is the check digit. The account number is therefore 126345678. Write a program to read in a nine digit integer from the keyboard and check it for these rules. Write"okay" or "not okay" for the results of the test.

**SOLUTION  
package Ques22;**

**import java.util.HashMap;**

**import java.util.Map;**

**import java.util.Scanner;**

**public class Main {**

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

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

**long number=sc.nextLong();**

**HashMap<Integer,Integer> map=new HashMap<Integer,Integer>();**

**int sum=0;**

**while(number>0)**

**{**

**int digit=(int)number%10;**

**number/=10;**

**if(!map.containsKey(digit))**

**map.put(digit,1);**

**else**

**map.put(digit,map.get(digit)+1);**

**sum+=digit;**

**}**

**boolean flag=false;**

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

**{**

**int tempSum=sum-entry.getKey();**

**int checkDigit=tempSum%10;**

**if(map.containsKey(checkDigit)) {**

**map.put(entry.getKey(), entry.getKey() - 1);**

**if (map.get(checkDigit) > 0) {**

**flag = true;**

**break;**

**}**

**else**

**map.put(entry.getKey(), entry.getKey() - 1);**

**}**

**sum+= entry.getKey();**

**}**

**if(flag)**

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

**else**

**System.out.println("not okay");**

**}**

**}**

**Java Advance Assignment Questions**

Q1. Write a java program as per the below given specification:

i) Design a Dish class having dishID, dishName, creationTime as the attributes. To initialize the attributes of the class and create a parameterized constructor. Also override the toString() method to print the object's details on the console.

ii) Design a Menu class having menuList as an attribute which is a static ArrayList. Load the menuList with a Dish object comprising some sample dishes using a static block.

iii) Design a Restaurant class comprising the main method. It should give a menu driven interface to a user as follows:

Press 1 to display Dishes.

Press 2 to search Dish.

iv) On the click of 1 all the dishes offered by the restaurant should be displayed to a user.

v) On the click of 2 it prompts the user to enter the dishID, if the ID entered matches with the dish, its details should be printed on the console.

**SOLUTION**

**package AQues1;**

**public class Dish {**

**private long dishID;**

**private String dishName;**

**private String time;**

**public Dish(long dishID, String dishName, String time) {**

**this.dishID = dishID;**

**this.dishName = dishName;**

**this.time = time;**

**}**

**public long getDishID() {**

**return dishID;**

**}**

**public String getDishName() {**

**return dishName;**

**}**

**public String getTime() {**

**return time;**

**}**

**@Override**

**public String toString() {**

**return "Dish{" +**

**"dishID=" + dishID +**

**", dishName='" + dishName + '\'' +**

**", time='" + time + '\'' +**

**'}';**

**}**

**}**

**package AQues1;**

**import java.util.ArrayList;**

**public class Menu {**

**static ArrayList<Dish> menulist=new ArrayList<Dish>();**

**static**

**{**

**menulist.add(new Dish(1,"Kadhai Paneer","17-Oct-2021"));**

**menulist.add(new Dish(2,"Dal Makhni","16-Oct-2021"));**

**menulist.add(new Dish(3,"Pizza","15-Oct-2021"));**

**menulist.add(new Dish(4,"Pasta","14-Oct-2021"));**

**}**

**}**

**package AQues1;**

**import java.util.Scanner;**

**public class Restaurant {**

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

**System.out.println("press 1 to display dishes");**

**System.out.print("press 2 to search dish : ");**

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

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

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

**Menu menu=new Menu();**

**if(x==1)**

**{**

**int i=1;**

**for(Dish dish:menu.menulist)**

**{**

**System.out.println("Dish "+i+" "+dish.getDishName());**

**i++;**

**}**

**}**

**else if(x==2)**

**{**

**System.out.print("Enter dishID : ");**

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

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

**for(Dish dish:menu.menulist)**

**{**

**if(checkID==dish.getDishID())**

**{**

**System.out.println("DishID : "+dish.getDishID());**

**System.out.println("DishName : "+dish.getDishName());**

**System.out.println("DishTime : "+dish.getTime());**

**break;**

**}**

**}**

**}**

**else**

**{**

**System.out.println("enter correct option");**

**}**

**}**

**}**

Q2. Write a java program to maintain a Phone book using Map. Program should give a menu driven interface as shown below:

Press 1 to Add new phone book entry

Press 2 to Search a Phone Number

Press 3 to Quit.

On the click of 1 a user is prompted to enter his name and phone number. The details entered by the user should be maintain in a Map phonebook, where the name is the key and the phone number is the value.

On the click of 2 a user is prompted to enter the name of the user whose number needs to be searched. Once the name entered correctly his number is displayed on the console.

On the click of 3 programs terminates.

Use HashMap to store phone book entries.

**SOLUTION  
package AQues2;**

**import sun.awt.AWTAccessor;**

**import java.util.HashMap;**

**import java.util.Map;**

**import java.util.Scanner;**

**public class Main {**

**static private Map<String,String> phoneBook=new HashMap<String,String>();**

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

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

**while(true) {**

**System.out.println("Press 1 to Add new phone book entry");**

**System.out.println("Press 2 to search a phone number");**

**System.out.println("Press 3 to Quit");**

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

**if(choice==1)**

**{**

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

**String tempName=sc.next();**

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

**String tempNumber=sc.next();**

**if(!phoneBook.containsKey(tempName))**

**phoneBook.put(tempName,tempNumber);**

**else**

**System.out.println("Number already present");**

**}**

**else if(choice==2)**

**{**

**System.out.println("Enter Name in Phone Book to search : ");**

**String tempName=sc.next();**

**if(phoneBook.containsKey(tempName))**

**{**

**System.out.println("Name : "+tempName+" - "+phoneBook.get(tempName));**

**}**

**}**

**else if(choice==3)**

**break;**

**}**

**}**

**}**

Q3.Create and Run a Thread using Runnable Interface and Thread class.

Use sleep and join methods with thread.

Use a singleThreadExecutor to submit multiple threads.

Try shutdown() and shutdownNow() and observe the difference.

Use isShutDown() and isTerminated() with ExecutorService.

Return a Future from ExecutorService by using callable and use get(), isDone(), isCancelled() with the Future object to know the status of the task submitted.

Submit List of tasks to ExecutorService and wait for the completion of all the tasks.

Schedule task using schedule(), scheduleAtFixedRate() and scheduleAtFixedDelay()

Increase concurrency with Thread pools using newCachedThreadPool() and newFixedThreadPool().

Use the Synchronize method to enable synchronization between multiple threads trying to access the method at same time.

Use the Synchronize block to enable synchronization between multiple threads trying to access the method at same time.

Use Atomic Classes instead of Synchronize methods and blocks.

Coordinate 2 threads using wait() and notify().

Coordinate multiple threads using wait() and notifyAll()

Use Reentrantlock for coordinating 2 threads with signal(), signalAll() and wait().

Create a deadlock and Resolve it using tryLock().

Q4. Create an application that, in turn, accesses each record in the lowerclassman file and then in the upperclassman file created in the LowerAndUpper application. Display an appropriate heading before

each student.

Q5.Write a program WordCount that reads a file and reports how many lines, words, and characters appear in it. Suppose, for example, that the file lear.txt contains the following passage from Shakespeare’s King Lear:

Poor naked wretches, wheresoe'er you are,

That bide the pelting of this pitiless storm,

How shall your houseless heads and unfed sides,

Your loop'd and window'd raggedness, defend you

From seasons such as these? O, I have ta'en

Too little care of this!

Given this file, your program should be able to generate the following sample run:

File: lear.txt

Lines = 6

Words = 47

Chars = 248

**SOLUTION  
package AQues5;**

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

**public class Main {**

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

**File file=new File("/home/knoldus/IdeaProjects/Assignment/src/AQues5/test.txt");**

**BufferedReader br=new BufferedReader(new FileReader(file));**

**String st;**

**int lines=0,words=0,characters=0;**

**while((st=br.readLine())!=null)**

**{**

**characters+=st.length();**

**String[] word=st.split("\\s");**

**words+=word.length;**

**lines++;**

**}**

**System.out.println("Line = "+lines+"\nWords = "+words+"\nCharacters = "+characters);**

**}**

**}**

Q6. Write a program that reads a list of exam scores from the file MidtermScores.txt (which contains one score per line) and then displays a histogram of those numbers, divided into the ranges 0–9, 10–19, 20–29, and so forth, up to the range containing only the value 100.

If, for example, MidtermScores.txt contains the data shown in the right margin, your program should then be able to generate a histogram that looks as much as possible like the following sample run:

00-09 :

10-19 : \*

20-29 : \*

30-39 : \*\*

40-49 : \*

50-59 : \*\*\*\*\*

60-69 : \*\*\*\*\*\*\*

70-79 : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

80-89 : \*\*\*\*\*\*

90-99 : \*\*\*\*\*\*\*\*\*\*

100 : \*  
**SOLUTION  
package AQues6;**

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

**public class Main {**

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

**File file=new File("/home/knoldus/IdeaProjects/Assignment/src/AQues6/midTermScores.txt");**

**BufferedReader br=new BufferedReader(new FileReader(file));**

**String st;**

**int range0to9=0,range10to19 = 0,range20to29 = 0,range30to39 = 0,range40to49 = 0,range50to59 = 0,range60to69 = 0,range70to79 = 0,range80to89 = 0**

**,range90to99 = 0,range100=0;**

**while((st=br.readLine())!=null)**

**{**

**int x=Integer.parseInt(st);**

**if(x>=0 && x<=9)**

**range0to9++;**

**else if(x>=10 && x<=19)**

**range10to19++;**

**else if(x>=20 && x<=29)**

**range20to29++;**

**else if(x>=30 && x<=39)**

**range30to39++;**

**else if(x>=40 && x<=49)**

**range40to49++;**

**else if(x>=50 && x<=59)**

**range50to59++;**

**else if(x>=60 && x<=69)**

**range60to69++;**

**else if(x>=70 && x<=79)**

**range70to79++;**

**else if(x>=80 && x<=89)**

**range80to89++;**

**else if(x>=90 && x<=99)**

**range90to99++;**

**else if(x==100)**

**range100++;**

**}**

**System.out.print("00 to 09 : ");**

**for(int i=0;i<range0to9;i++)**

**System.out.print("\*");**

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

**System.out.print("10 to 19 : ");**

**for(int i=0;i<range10to19;i++)**

**System.out.print("\*");**

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

**System.out.print("20 to 29 : ");**

**for(int i=0;i<range20to29;i++)**

**System.out.print("\*");**

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

**System.out.print("30 to 39 : ");**

**for(int i=0;i<range30to39;i++)**

**System.out.print("\*");**

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

**System.out.print("40 to 49 : ");**

**for(int i=0;i<range40to49;i++)**

**System.out.print("\*");**

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

**System.out.print("50 to 59 : ");**

**for(int i=0;i<range50to59;i++)**

**System.out.print("\*");**

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

**System.out.print("60 to 69 : ");**

**for(int i=0;i<range60to69;i++)**

**System.out.print("\*");**

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

**System.out.print("70 to 79 : ");**

**for(int i=0;i<range70to79;i++)**

**System.out.print("\*");**

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

**System.out.print("80 to 89 : ");**

**for(int i=0;i<range80to89;i++)**

**System.out.print("\*");**

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

**System.out.print("90 to 99 : ");**

**for(int i=0;i<range90to99;i++)**

**System.out.print("\*");**

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

**System.out.print("100 : ");**

**for(int i=0;i<range100;i++)**

**System.out.print("\*");**

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

**}**

**}**

Q7. Write and explain the output of the following program?

public class JavaHungry {

public static void main(String args[])

{

try

{

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

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

{

System.out.print ("Array elements are : " + arr[i] + "\n");

}

}

catch (Exception e)

{

System.out.println ("Exception : " + e);

}

catch (ArrayIndexOutOfBoundsException ex)

{

System.out.println ("ArrayIndexOutOfBoundsException : "+ ex);

}

}

}.  
**SOLUTION  
The above program won’t run because the first catch statement is catching an exception of a parent class before than a sub - class, so best practice to write multiple catch statements in exception handling is to catch specific exceptions at first and than at the last we should catch exception of a parent class i.e the Exception class.**

Q8. WAP to handle the exception using try and multiple catch blocks.  
**SOLUTION  
package AQues8;**

**import java.util.InputMismatchException;**

**public class Main {**

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

**boolean flag=false;**

**try**

**{**

**int x=1000/0;**

**}**

**catch(ArrayIndexOutOfBoundsException e)**

**{**

**flag=true;**

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

**e.printStackTrace();**

**}**

**catch(InputMismatchException e)**

**{**

**flag=true;**

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

**e.printStackTrace();**

**}**

**catch(ArithmeticException e)**

**{**

**flag=true;**

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

**e.printStackTrace();**

**}**

**catch(Exception e)**

**{**

**flag=true;**

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

**e.printStackTrace();**

**}**

**finally**

**{**

**if(flag)**

**System.out.println("Exception catched");**

**else**

**System.out.println("No Exceptions catched");**

**}**

**}**

**}**

Q9. Create a class TwoDim which contains private members as x and y coordinates in package P1. Define the default constructor, a parameterized constructor and override toString() method to display the coordinates. Now reuse this class and in package P2 create another class ThreeDim, adding a new dimension as z as its private member. Define the constructors for the subclass and override toString() method in the subclass also. Write appropriate methods to show dynamic method dispatch. The main() function should be in a package P.  
**SOLUTION  
package AQues9.P1;**

**public class TwoDim {**

**private double x;**

**private double y;**

**public TwoDim(double x, double y)**

**{**

**this.x=x;**

**this.y=y;**

**}**

**@Override**

**public String toString() {**

**return "TwoDim{" +**

**"x=" + x +**

**", y=" + y +**

**'}';**

**}**

**}**

**package AQues9.P2;**

**import AQues9.P1.TwoDim;**

**public class ThreeDim extends TwoDim {**

**private double z;**

**public ThreeDim(double x, double y, double z)**

**{**

**super(x,y);**

**this.z=z;**

**}**

**@Override**

**public String toString() {**

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

**return "ThreeDim{" +**

**"z=" + z +**

**'}';**

**}**

**}**

**package AQues9.P;**

**import AQues9.P1.TwoDim;**

**import AQues9.P2.ThreeDim;**

**public class Main**

**{**

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

**{**

**TwoDim obj=new TwoDim(2.3,4.5);**

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

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

**System.out.println("Dynamic method dispatch");**

**obj=new ThreeDim(2.3,4.5,5.6);**

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

**}**

**}**

Q10. Is the below code written correctly? If not then give reason.

class A

{

String s = "AAA";

void methodA()

{

System.out.println(s);

}

static class B

{

void methodB()

{

methodA();

}

}

}

**SOLUTION  
The code above is not written correctly as a non-static reference cannot be referenced from a static context and static inner classes can only access static members of the outer class.  
  
CORRECT CODE**   
**class A**

**{**

**static String s = "AAA";**

**static void methodA()**

**{**

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

**}**

**static class B**

**{**

**void methodB()**

**{**

**methodA();**

**}**

**}**

**}**

Q11. Write the output of the following code.

class Outer {

void outerMethod() {

System.out.println("Inside outerMethod");

// Inner class is local to outerMethod()

class Inner {

void innerMethod() {

System.out.println("Inside innerMethod");

}

}

Inner y = new Inner();

y.innerMethod();

}

}

class Test1 {

public static void main(String[] args) {

Outer x = new Outer();

x.outerMethod();

}

}.

**OUTPUT**

**Inside outerMethod**

**Inside innerMethod**

Q12. Design an Employee class having attributes as empID, empName & empAge. The sample data of Employees is stored in a text file Employee.txt w here the details are separated by “,”. Load an array of Employee using static block which comprises the Employee object created from the sample data stored in Employee.txt file. Allow the user to modify the Employee details in the array and save the updated information back to the file Employee.txt.  
**SOLUTION  
  
EMPLOYEE CLASS  
package AQues12;**

**public class Employee {**

**private int empId;**

**private String empName;**

**private int age;**

**public Employee(int empId, String empName, int age) {**

**this.empId = empId;**

**this.empName = empName;**

**this.age = age;**

**}**

**public int getEmpId() {**

**return empId;**

**}**

**public String getEmpName() {**

**return empName;**

**}**

**public int getAge() {**

**return age;**

**}**

**@Override**

**public String toString() {**

**return "Employee{" +**

**"empId=" + empId +**

**", empName='" + empName + '\'' +**

**", age=" + age +**

**'}';**

**}**

**}**

**MAIN CLASS  
package AQues12;**

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

**import java.util.Scanner;**

**public class Main**

**{**

**static Employee[] arr;**

**static {**

**arr=new Employee[1];**

**}**

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

**File file = new File("/home/knoldus/IdeaProjects/Assignment/src/AQues12/Employee.txt");**

**//READING FROM FILE**

**BufferedReader br = new BufferedReader(new FileReader(file));**

**String st;**

**System.out.println("Employee Details Currently inside Employee.txt: ");**

**while((st=br.readLine())!=null)**

**{**

**String[] temp=st.trim().split("\\,");**

**//STORING INSIDE EMPLOYEE OBJECT**

**Employee emp=new Employee(Integer.parseInt(temp[0]),temp[1],Integer.parseInt(temp[2]));**

**arr[0]=emp;**

**}**

**br.close();**

**//DISPLAYING OBJECT CONTENT**

**System.out.println(arr[0]);**

**//WRITING NEW CONTENT TO FILE**

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

**System.out.println("\nEnter new details of Employee to be present inside Employee.txt: ");**

**System.out.print("Enter ID : ");**

**String id=sc.nextLine();**

**System.out.print("\nEnter Name : ");**

**String name=sc.nextLine();**

**System.out.print("\nEnter Age : ");**

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

**String text=id+","+name+","+age;**

**FileWriter fw=new FileWriter(file);**

**fw.write(text);**

**fw.flush();**

**fw.close();**

**//DISPLAYING NEW CONTENT OF Employee.txt**

**br = new BufferedReader(new FileReader(file));**

**System.out.println("\nNew Employee Details inside Employee.txt: ");**

**while((st=br.readLine())!=null)**

**{**

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

**}**

**}**

**}**

Q13. Write a Java program that implements a multi-thread application that has three threads. First thread generates a random integer every 1 second and if the value is even, the second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of the cube of the number.  
  
**SOLUTION  
package AQues13;**

**import java.util.Random;**

**import java.util.concurrent.Executors;**

**import java.util.concurrent.ScheduledExecutorService;**

**import java.util.concurrent.TimeUnit;**

**class Square implements Runnable**

**{**

**int x;**

**public Square(int x) {**

**this.x = x;**

**}**

**public void run() {**

**System.out.println("Square is "+x\*x);**

**}**

**}**

**class Cube implements Runnable**

**{**

**int x;**

**public Cube(int x) {**

**this.x = x;**

**}**

**public void run() {**

**System.out.println("Cube is "+x\*x\*x);**

**}**

**}**

**public class Main {**

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

**ScheduledExecutorService exec = Executors.newSingleThreadScheduledExecutor();**

**exec.scheduleAtFixedRate(new Runnable() {**

**@Override**

**public void run() {**

**Random random=new Random();**

**int n=random.nextInt(1000);**

**System.out.println("Random Integer Generated : "+n);**

**if(n%2==0)**

**{**

**Square obj=new Square(n);**

**obj.run();**

**}**

**else**

**{**

**Cube obj=new Cube(n);**

**obj.run();**

**}**

**}**

**}, 0, 1, TimeUnit.SECONDS);**

**}**

**}**

Q14. Write a program having user interface like

4. accept first name and surname

5. display total name

6. exit

Option A should accept First Name and SurName from command prompt and save that to Vector object

Option B it has to display how many names entered in the vector object

This menu should be repeated till users select exit.

To store first name and surname, create a class Name with these two attributes.

**SOLUTION**

**NAME CLASS**

**package AQues14;**

**public class Name {**

**private String firstName;**

**private String lastName;**

**public Name(String firstName, String lastName) {**

**this.firstName = firstName;**

**this.lastName = lastName;**

**}**

**public String getFirstName() {**

**return firstName;**

**}**

**public String getLastName() {**

**return lastName;**

**}**

**@Override**

**public String toString() {**

**return "Name{" +**

**"firstName='" + firstName + '\'' +**

**", lastName='" + lastName + '\'' +**

**'}';**

**}**

**}**

**MAIN CLASS  
  
 package AQues14;**

**import java.util.List;**

**import java.util.Scanner;**

**import java.util.Vector;**

**public class Main {**

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

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

**List<Name> names=new Vector<Name>();**

**while(true)**

**{**

**System.out.println("<----Press one of the following options---->");**

**System.out.println("4. accept first name and surname");**

**System.out.println("5. display total name");**

**System.out.println("6. exit");**

**System.out.print("Enter option number : ");**

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

**if(choice==4)**

**{**

**System.out.print("Enter FirstName : ");**

**String firstName=sc.nextLine();**

**System.out.print("Enter LastName : ");**

**String lastName=sc.nextLine();**

**Name tempName=new Name(firstName,lastName);**

**names.add(tempName);**

**}**

**else if (choice==5)**

**{**

**if(names.size()==0)**

**{**

**System.out.println("No names inside Vector");**

**}**

**else {**

**System.out.println("All the names entered are : ");**

**for (Name name : names) {**

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

**}**

**}**

**}**

**else if(choice==6)**

**break;**

**else**

**System.out.println("Enter correct option");**

**}**

**}**

**}**

Q15. Write a Java program to get the last modified time of a file.   
**SOLUTION  
package AQues15;**

**import java.io.File;**

**import java.text.DateFormat;**

**import java.text.SimpleDateFormat;**

**public class Main {**

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

**File file =new File("/home/knoldus/IdeaProjects/Assignment/src/AQues15/LastModifiedTime.txt");**

**DateFormat sdf = new SimpleDateFormat("MMMM dd, yyyy hh:mm a");**

**System.out.println("Last Modified Time "+sdf.format(file.lastModified()));**

**}**

**}**