**1Q calling method with no return and parameter ?**

**public** **class** Swap1 {

**public** **void** swap (**int** a, **int** b)

{

**int** temp = a;

a = b;

b = temp;

}

}

**class** SwapTester{

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

Swap1 S = **new** Swap1();

S.swap(6,4);

System.***out***.println("successfully swapped");

}

}

**2Q Calling method with return and no parameter?**

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

**private** **double** balance;

**public** BankAccount(**double** initialBalance)

{

balance = initialBalance;

}

**public** **double** getBalance(){

**return** balance;

}

}

**public** **class** BankAccountTester {

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

BankAccount ba1 = **new** BankAccount(1000);

System.***out***.println("The balance of account is :" + ba1.getBalance());

System.***out***.println("should be 1000.");

}

}

**3Q Calling method with return and parameter ?**

**public** **class** MinimumNum {

**int** answer;

**public** **int** min (**int** a, **int** b)

{

**if** (a <= b)

answer = a;

**else**

answer = b;

**return** answer;

}

}

**class** MinTester{

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

MinimumNum N = **new** MinimumNum();

System.***out***.println(N.min(6,7));

}

}

**4Q calling method with void**

**public** **class** BankAcc {

**public** **double** balance;

**public** BankAcc(**double** initialBalance)

{

balance = initialBalance;

}

**public** **void** deposit(**double** amount)

{

balance = balance + amount;

}

}

**class** BankTester {

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

BankAcc ba1 = **new** BankAcc(1000);

ba1.deposit(1000);

System.***out***.println("The balance of account is :" + ba1.balance);

}

}

**5Q calling static method**

**public** **class** StaticM {

**public** **static** **void** method(){

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

}

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

StaticM.*method*();

}

}

**6Q create classes under multiple packages**

package concept;

public class Package1Class {

}

package theory;

public class Package2Class {

//look that there is not use of the import statement, however everytime you need

//to call Package1Class, you should provide the full package name

private concept.Package1Class x;

public Package2Class (package1.Package1Class x) {

this.x = x

}

}

**7Q Create default/paramterzied constructors**

**public** **class** Student {

**int** id;

String name;

Student(){

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

}

Student(**int** i,String n){

id = i;

name = n;

}

**void** display()

{

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

}

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

Student s = **new** Student();

Student s1 = **new** Student(111,"mahi");

s.display();

s1.display();

}

}

**8Q create Employee class**

Go to Project 🡪 project name 🡪 finish🡪 new 🡪 class 🡪 Employee

**9Q**  **create .java file/class**

File -------> new---> class file---> Java.class file should always begin with upper case letters

**10Q create method that return list of employee collection**

**import** java.util.ArrayList;

**import** java.util.List;

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

**int** empID;

String empName;

**int** empAge;

**public** Employee(**int** empID,String empName,**int** empAge)

{

**this**.empID=empID;

**this**.empName=empName;

**this**.empAge=empAge;

}

**public** **int** getEmpId() {

**return** empID;

}

**public** **void** setEmpid(**int** empid) {

**this**.empID = empid;

}

**public** String getname() {

**return** empName;

}

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

**this**.empName = name;

}

}

**class** EmployeeTester {

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

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

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

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

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

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

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

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

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

}

}

}

**11Q create static block**

**public** **class** StaticBlock {

**static** {

System.***out***.println("This is first static block");

}

**public** **static** **void** staticMethod() {

System.***out***.println("This is static method");

}

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

StaticBlock.*staticMethod*();

}

}

**12Q creating method with return data type and parameter**

**public** **int** min (**int** a, **int** b)

{

**int** answer;

**if** (a <= b)

answer = a;

**else**

answer = b;

**return** answer;

}

**13Q creating method with return data type, we can return int/string/double/float/date etc**

|  |
| --- |
| yes we can return a value from any of the specific method |
| Note: value that we specify after return keyword should be of data type that is specified in  method signature |

**14Q creating method with void**

**public** **void** deposit(**double** amount)

{

balance = balance + amount;

}

**15Q creating object**

**public** **class** Point {

**public** **int** x = 0;

**public** **int** y = 0;

//constructor

**public** Point(**int** a, **int** b) {

x = a;

y = b;

}

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

Point p = **new** Point(24, 35);

}

}

**16Q creating property/data members :**

**public** **class** Square {

**private** **double** side;

**public** Square(**double** s) {

side = s;

}

// This is a read property

**public** **double** getSide() {

**return** side;

}

**double** calculateArea() {

**return** side \* side;

}

**17Q creating static method**

**public** **static** **void** method(){

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

}

**18Q creating static property:**

**public** **class** StaticBlock {

**static** {

System.***out***.println("This is first static block");

}

**static** String *stringName* = "static variable";

**public** **static** **void** staticMethod() {

System.***out***.println("This is static method and "+ *stringName*);

}

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

StaticBlock.*staticMethod*();

}

}

**19Q creating variable, we can create variables inside method**

we can create variable using data types and we can also create variables inside the method as local variables.

**20Q Difference between string, string buffer, string builder with example**

**String:** string is a data type, it is immutable

Ex: string b = “Hello”;

**StringBuffer:** it is mutable and thread safe

**String Builder:** it is mutable and faster than string buffer . but it is not thread safe

**public** **class** StringBuild {

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

StringBuilder sBuilder = **new** StringBuilder();

sBuilder.append(" String Buffer");

sBuilder.append(" Buffer");

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

}

}

**21Q how to create packages and what is best way to give name**

file---->new ----> package.Packages should always begin with lower case letters

**22Q How to do inheritance in java (using extend keyword)**

**public** **class** Account {

**private** String accType;

**private** **long** accNum;

**private** **long** amount;

Account(String accType, **long** amount , **long** accNum)

{

**this**.accType = accType;

**this**.accNum = accNum;

setAmount(amount);

}

**public** **void** deposit(**long** amount)

{

**this**.amount += amount;

}

**public** String getAccType()

{

**return** accType;

}

**public** **long** getAccNum()

{

**return** accNum;

}

**public** **long** getAmount()

{

**return** amount;

}

**public** **void** setAmount(**long** amount)

{

**this**.amount = amount;

}

}

**class** SavingsAccount **extends** Account

{

SavingsAccount(**long** amount)

{

**super**("savings Account", amount, 136511122);

}

}

**class** AccountTester

{

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

{

SavingsAccount mahi = **new** SavingsAccount(10000);

System.***out***.println("account Type: " + mahi.getAccType());

System.***out***.println("account Number: " + mahi.getAccNum());

System.***out***.println("initial amount: " + mahi.getAmount());

mahi.deposit(5000);

System.***out***.println("new amount after deposit: " + mahi.getAmount());

}

}

**24Q implementing interface**

**public** **interface** Animal {

**public** **void** eat();

**public** **void** travel();

}

**class** MammalInt **implements** Animal {

**public** **void** eat() {

System.***out***.println("Mammal eats");

}

**public** **void** travel() {

System.***out***.println("Mammal travels");

}

**public** **int** noOfLegs() {

**return** 0;

}

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

MammalInt m = **new** MammalInt();

m.eat();

m.travel();

}

}

**25Q implementing polymorphism**

**public** **class** Employee1 {

**private** String name;

**private** String address;

**public** Employee1(String name, String address) {

System.***out***.println("Constructing an Employee");

**this**.name = name;

**this**.address = address;

}

**public** **void** mailCheck() {

System.***out***.println("Mailing a check to " + **this**.name + " " + **this**.address);

}

**public** String toString() {

**return** name + " " + address;

}

**public** String getName() {

**return** name;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String newAddress) {

address = newAddress;

}

}

**class** Salary **extends** Employee1 {

**private** **double** salary; // Annual salary

**public** Salary(String name, String address, **double** salary) {

**super**(name, address);

setSalary(salary);

}

**public** **void** mailCheck() {

System.***out***.println("Within mailCheck of Salary class ");

System.***out***.println("Mailing check to " + getName()

+ " with salary " + salary);

}

**public** **double** getSalary() {

**return** salary;

}

**public** **void** setSalary(**double** newSalary) {

**if**(newSalary >= 0.0) {

salary = newSalary;

}

}

}

**class** VirtualDemo {

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

Salary s = **new** Salary("Mahi", "hyd, Ap", 3600.00);

Employee1 e = **new** Salary("chinu", "bombay, MH", 2400.00);

System.***out***.println("Call mailCheck using Salary reference --");

s.mailCheck();

System.***out***.println("\n Call mailCheck using Employee reference--");

e.mailCheck();

}

}

**26Q implement method overloading**

**public** **class** Concept1 {

**public** **void** display(**int** a){

System.***out***.println("The number is : " + a );

}

**public** **void** display(**int** a , **int** b){

System.***out***.println("The numbers are : " + a + " " + b);

}

}

**class** OverLoading{

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

Concept1 obj = **new** Concept1();

obj.display(10);

obj.display(10,20);

}

}

**27Q implement method overriding**

**public** **class** Vehicle {

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

{

System.***out***.println("Vehicle is running");

}

}

**class** Car **extends** Vehicle{

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

{

System.***out***.println("Car is running safely");

}

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

Car C1 = **new** Car();

C1.run();

}

}

**28Q method that will return hard coded value**

Hard coding is not only part of java. It actually comes when we put the original variables and data values In place of hard coding

**29Q method that will return property value**

**Repeated**

**30Q Steps to create project**

Open eclipse 🡪 file 🡪 project🡪 java project🡪 project name 🡪 select environment 🡪 finish

**31Q Steps to create workspace**

After installing eclipse .open eclipse .it will ask path for workspace, then browse the files and setup workspace path.

**32Q Steps to install eclipse**

Step 0: Install JDK

To use Eclipse for Java programming, you need to first install Java Development Kit (JDK).

Step 1: Download

Download Eclipse from https://www.eclipse.org/downloads. Under "Get Eclipse Neon" ⇒ Click "Download Packages". For beginners, choose the 3rd entry "Eclipse IDE for Java Developers" (32-bit or 64-bit) (e.g., "eclipse-java-neon-2-win32-x86\_64.zip" 161MB) ⇒ Download.

Step 2: Unzip

To install Eclipse, simply unzip the downloaded file into a directory of your choice (e.g., "d:\myproject").

There is no need to run any installer. Moreover, you can simply delete the entire Eclipse directory when it is no longer needed (without running any un-installer). You are free to move or rename the directory. You can install (unzip) multiple copies of Eclipse in the same machine.

**33Q Steps to install java**

Step 1: Download JDK

Goto Java SE download site @ http://www.oracle.com/technetwork/java/javase/downloads/index.html.

Under "Java Platform” ⇒ Click the "JDK Download" button.

Look for the latest "Java SE Development Kit 8u{xx}" ⇒ Check "Accept License Agreement".

Choose the JDK for your operating system,.

Step 2: Install JDK and JRE

Run the downloaded and install files.

**34Q what is checked exception/unchecked exception**

**checked exception:** The program throw exception at compilation time . it should be handled by trycatch block.

**unchecked exception:** The program does not imply any exception during compilation time. But it throws argument exception during runtime

**35Q what is data type and different data types**

They are used to classify the value of a variable in a programming language**.** They are different types like boolean ,int ,char ,double ,long ,float , short

**36Q what is final keyword, create final class, final method, final property`**

If we use final keyword to any class, method or variable .we cannot override it

**final** **class** Final {

//Blank final variable

**final** **int** MAX\_VALUE;

Final(){

//It must be initialized in constructor

MAX\_VALUE=100;

}

**final** **void** myMethod(){

System.***out***.println(MAX\_VALUE);

}

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

Final obj=**new** Final();

obj.myMethod();

}

}

**37Q what is main method will do?**

All Java applications begin processing with a main() method;

Each statement in the main executes in order until the end of main is reached -- this is when your program terminates;

**38Q What is variable?**

A variable provides us with named storage that our programs can manipulate

**39Q write code for creating abstract class**

**public** **class** Abstraction {

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

TwoWheeler test = **new** Honda();

test.run();

}

}

**abstract** **class** TwoWheeler {

**public** **abstract** **void** run();

}

**class** Honda **extends** TwoWheeler{

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

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

}

}

**40Q write code for interface and create class to implement that interface**

**Repeated question**

**41Q write code to add items to ArrayList collection**

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

**public** **class** ArrayListExample {

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

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

ArrayList<String> frnds = **new** ArrayList<String>();

frnds.add("mahi");

frnds.add("divya");

frnds.add("alekya");

frnds.add("varma");

System.***out***.println("Currently the array list has following elements:"+ frnds);

}

}

**42Q write code to add items to integer, string array**

public static void main(String[] args) {

int[] series = {4,2};

series = addElement(series, 3);

series = addElement(series, 1);

}

static int[] addElement(int[] a, int e) {

a = Arrays.copyOf(a, a.length + 1);

a[a.length - 1] = e;

return a;

}

**43Q write code to handle exceptions with try/catch/finally**

public void openFile() throws IOException {

FileReader reader = null;

try {

reader = new FileReader("someFile");

int i=0;

while(i != -1){

i = reader.read();

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

}

} finally {

if(reader != null){

try {

reader.close();

} catch (IOException e) {

//do something clever with the exception

}

}

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

}

}

**44Q write code to retrieve items from arraylist (using for each loop**

p**ublic** **class** AList {

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

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

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

list.add("mahija");

list.add("divya");

list.add("harshi");

list.add("hari");

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

System.***out***.println("Name: "+list.get(i)+" ");

}

}

}

**45Q write code to retrieve items from integer, string array**

**public** **class** Medals {

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

**final** **int** COUNTRIES = 7;

**final** **int** MEDALS = 3;

String[] countries =

{

"Canada",

"China",

"Germany",

"Korea",

"Japan",

"Russia",

"United States"

};

**int** [][] counts = {

{ 1, 0, 1 },

{ 1, 1, 0 },

{ 0, 0, 1 },

{ 1, 0, 0 },

{ 0, 1, 1 },

{ 0, 1, 1 },

{ 1, 1, 0 }

};

System.***out***.println(" country Gold Silver Bronze Total");

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

System.***out***.printf("%15s", countries[i]);

**int** total = 0;

**for** (**int** j =0; j< MEDALS; j++){

System.***out***.printf("%8d", counts[i][j]);

total = total + counts [i][j];

}

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

System.***out***.printf("%8d\n", total);

}

}

}

**46Q Write method to return list of rows code to loop throughs**

p**ublic** **class** AList {

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

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

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

list.add("mahija");

list.add("divya");

list.add("harshi");

list.add("hari");

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

System.***out***.println("Name: "+list.get(i)+" ");

}

}

}