**JAVA CONCEPTS**

1. Add employee class to list collection

package concepts;

import java.util.ArrayList;

import java.util.List;

public class EmployeeClass {

int empID;

String empName;

int empAge;

public EmployeeClass(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;

}

public int getEmpAge() {

return empAge;

}

public void setEmpAge(int empAge) {

this.empAge = empAge;

}

}

class EmployeeTester {

public static void main(String[] args) {

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

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

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

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

list.add(new EmployeeClass(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()+ " ");

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

}

}

}

2. Calling Classes under different packages

package package1;

import package2.Class2;

public class Class1 {

public void method1(){

System.*out*.println("This is method from package1");

}

public static void main (String[] args){

Class2 obj2 = new Class2();

obj2.method2();

}

}

package package2;

import package1.Class1;

public class Class2 {

public void method2(){

System.*out*.println("This is method from package2");

}

public static void main (String[] args){

Class1 obj1 = new Class1();

obj1.method1();

}

}

3. Calling method with no return type and parameter

package concepts;

public class Task3 {

public void method(String a, String b){

System.*out*.println("This is a method with ==> " + a + " and " +b);

}

public static void main(String[] args) {

Task3 obj = new Task3();

obj.method("no return", "with parameter");

}

}

4. Calling method with return type and no parameter

public class Task4 {

public int method(){

int a = 10;

int b = 10;

int c = a+b;

System.*out*.println("This is a method with return type but no parameters");

System.*out*.println("Value of c ==> " + c);

return c;

}

public static void main(String[] args){

Task4 obj = new Task4();

obj.method();

}

}

5. Calling methods with return type and parameters.

package concepts;

public class Task5 {

public int method(int a, int b){

int c = a+b;

System.*out*.println("This is a method with return type and parameters");

System.*out*.println("Value of c ==> " + c);

return c;

}

public static void main(String[] args){

Task5 obj = new Task5();

obj.method(10, 20);

}

}

6. Calling method with return and storing the return value.

7. Calling method with void

package concepts;

public class Task7 {

public void method(){

System.*out*.println("A method with no return type and no parameter");

}

public static void main(String[] args){

Task7 obj = new Task7();

obj.method();

}

}

8. Calling Static method

package concepts;

public class Task8 {

public static void method(){

System.*out*.println("Calling static method");

}

public static void main(String[] args){

Task8.*method*();

}

}

9. Same as Task2

package package1;

import package2.Class2;

public class Class1 {

public void method1(){

System.*out*.println("This is method from package1");

}

public static void main (String[] args){

Class2 obj2 = new Class2();

obj2.method2();

}

}

package package2;

import package1.Class1;

public class Class2 {

public void method2(){

System.*out*.println("This is method from package2");

}

public static void main (String[] args){

Class1 obj1 = new Class1();

obj1.method1();

}

}

10. Create default or parameterized constructors

package concepts;

public class Task10 {

int a, b, c;

static int *d*;

public int result() {

*d* = a\*b\*c;

return *d*;

}

Task10(){

this.a = 1;

this.b = 1;

this.c = 10;

}

Task10(int a,int b,int c){

this.a = a;

this.b = b;

this.c = c;

}

public static void main(String[] args){

Task10 obj1 = new Task10();

Task10 obj2 = new Task10(2, 2, 2);

obj1.result();

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

System.*out*.println("Value is " + *d*);

obj2.result();

System.*out*.println("This is Parameterized constructor");

System.*out*.println("Value is " + *d*);

}

}

11. Create Employee class

package concepts;

public class Employee {

public static void main(String[] args){

System.*out*.println("This is Task11");

}

}

12. Create .java file/class

Click on File 🡪 New 🡪 Java Project

Right Click on Project 🡪 New 🡪 Class

13. Create method that returns list of employee collection

Refer 1

14. Create static block

package concepts;

public class Task14{

static {

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

}

public Task14(){

System.*out*.println("This is constructor");

}

public static String *variable* = "Static Variable";

static {

System.*out*.println("This is second static block and " + *variable*);

}

public static void main(String[] args){

Task14 obj = new Task14();

Task14.*staticMethod2*();

}

static {

*staticMethod*();

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

}

public static void staticMethod() {

System.*out*.println("This is static method 1");

}

public static void staticMethod2() {

System.*out*.println("This is static method 2");

}

}

15. Creating method with return data type and parameter.

Same as Task 5

package concepts;

public class Task5 {

public int method(int a, int b){

int c = a+b;

System.*out*.println("This is a method with return type and parameters");

System.*out*.println("Value of c ==> " + c);

return c;

}

public static void main(String[] args){

Task5 obj = new Task5();

obj.method(10, 20);

}

}

16. Creating method with return type, we can return int/string/float/double.

Same as Task 4

public class Task4 {

public int method(){

int a = 10;

int b = 10;

int c = a+b;

System.*out*.println("This is a method with return type but no parameters");

System.*out*.println("Value of c ==> " + c);

return c;

}

public static void main(String[] args){

Task4 obj = new Task4();

obj.method();

}

}

17. Creating object

Same as Task 7

package concepts;

public class Task7 {

public void method(){

System.*out*.println("A method with no return type and no parameter");

}

public static void main(String[] args){

Task7 obj = new Task7();

obj.method();

}

}

18. Creating property/data members.

package concepts;

public class Task18 {

int a=10;

int b=10;

public static void main(String args[]){

System.*out*.println("This is Task 18");

}

}

19. Creating static method

Same as Task 8

package concepts;

public class Task8 {

public static void method(){

System.*out*.println("Calling static method");

}

public static void main(String[] args){

Task8.*method*();

}

}

20. Creating Static property

Same as Task 10

package concepts;

public class Task10 {

int a, b, c;

static int *d*;

public int result() {

*d* = a\*b\*c;

return *d*;

}

Task10(){

this.a = 1;

this.b = 1;

this.c = 10;

}

Task10(int a,int b,int c){

this.a = a;

this.b = b;

this.c = c;

}

public static void main(String[] args){

Task10 obj1 = new Task10();

Task10 obj2 = new Task10(2, 2, 2);

obj1.result();

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

System.*out*.println("Value is " + *d*);

obj2.result();

System.*out*.println("This is Parameterized constructor");

System.*out*.println("Value is " + *d*);

}

}

21. Creating variables, we can create variables inside a method.

Same as Task 5

package concepts;

public class Task5 {

public int method(int a, int b){

int c = a+b;

System.*out*.println("This is a method with return type and parameters");

System.*out*.println("Value of c ==> " + c);

return c;

}

public static void main(String[] args){

Task5 obj = new Task5();

obj.method(10, 20);

}

}

22. Difference between String, String buffer, String builder with example.

The most important difference between String and StringBuffer/StringBuilder in java is that String object is immutable whereas StringBuffer/StringBuilder objects are mutable. By immutable, we mean that the value stored in the String object cannot be changed.

23. How to create packages, what is the best way to give the names

Click on File 🡪 New 🡪 Java Project

Right Click on Project 🡪 New 🡪 Package

24. Inheritance in JAVA

package concepts;

public class ParentClass {

static int *a* = 10;

static int *b* = 20;

}

package concepts;

public class ChildClass extends ParentClass{

public static void main(String[] args){

int c = *a*+*b*;

System.*out*.println("Value of c is " + c);

}

}

25. How to update the data into xml file and read data from xml file.

26. Implementing Interface.

package concepts;

public interface Task44interface {

int *length* = 10;

int *breadth* = 10;

int *height* = 10;

public abstract int method();

}

package concepts;

public class Task44class implements Task44interface {

public static void main(String[] args) {

Task44class obj = new Task44class();

obj.method();

}

@Override

public int method() {

int volume = *length*\**breadth*\**height*;

System.*out*.println("Volume is "+ volume);

return volume;

}

}

27. Implementing Polymorphism

Refer 28 and 29

28. Implementing Method Overloading

package concepts;

public class Overloading1 {

public void method1(int a, String b, int c){

System.*out*.println("Method1");

}

public void method1(String a, int b){

System.*out*.println("Method2");

}

}

package concepts;

public class Overloading2 {

public static void main(String[] args){

Overloading1 obj = new Overloading1();

obj.method1(10, "divya", 10);

System.*out*.println("Called method1 based on arguments");

obj.method1("divya", 20);

System.*out*.println("Called method2 based on arguments");

}

}

29. Implementing Method Overriding

package concepts;

public class Overriding1 {

public void method(){

System.*out*.println("This is Super Class");

}

}

package concepts;

public class Overriding2 extends Overriding1{

public void method(){

System.*out*.println("This is Sub Class");

}

}

package concepts;

public class Overriding3 {

public static void main(String[] args){

Overriding1 obj1 = new Overriding1();

obj1.method();

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

Overriding2 obj2 = new Overriding2();

obj2.method();

}

}

30. Method that will return hard coded value

package concepts;

public class Task30 {

public int hardCoded(){

int a =10;

int b = a +10;

System.*out*.println("Value is " +b);

return b;

}

public static void main(String[] args){

Task30 obj = new Task30();

obj.hardCoded();

}

}

31. Method that will return property value

package concepts;

public class Task31 {

public int method(int a, int b){

int c = a+b;

System.*out*.println("Value is "+ c);

return c;

}

public static void main(String[] args){

Task31 obj = new Task31();

obj.method(10,10);

}

}

32. Steps to create project

Click on File 🡪 New 🡪 Java Project

33. Steps to create work space

Open Eclipse 🡪Browse the location of the folder 🡪 Click OK

34. Steps to install eclipse

Unzip eclipse-standard-luna-R-win32.zip, the file that you just downloaded and moved.

Create a shortcut on your desktop to the eclipse.exe file in this eclipse folder.

Double-click the shortcut to Eclipse that you just created above.

Click OK.

35. Steps to install JAVA

Open the desktop, and then tap or click the Internet Explorer icon on the taskbar.

Go to Java.com.

Tap or click the Free Java Download button, and then tap or click Agree and Start Free Download.

On the notification bar, tap or click Run.

36. Using static property, it will maintain

Same as Task 10

37. What is checked exception and un-checked exception?

**Checked exceptions** are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throwskeyword.

**Unchecked** are the exceptions that are not checked at compiled time. In Java exceptions under Errorand RuntimeExceptionclasses are unchecked exceptions.

38. What is data type, different types of data types?

A particular kind of data item, as defined by the values it can take, the programming language used, or the operations that can be performed on it. The eight primitive data types are: byte, short, int, long, float, double, boolean, and char.

39. What is final keyword, create final class, final method, final property

**package** concepts;

**final** **class** Task39 {

**final** **int** a;

Task39(){

a=100;

}

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

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

}

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

Task39 obj=**new** Task39();

obj.myMethod();

}

}

40. What is main method will do?

In Java, main is a static method. This means the method is part of its class and not part of objects. This main method executes first when JVM starts.

41. What is variable?

A Java variable is a piece of memory that can contain a data value. A variable thus has a data type.

42. Write a code to save data into excel file and read from excel file (POI and jxcel)

43. Write code for creating abstract class.

package concepts;

public abstract class Task43 {

int length = 10;

int breadth = 10;

int height = 10;

public abstract int method();

}

package concepts;

public class Task43subclass extends Task43{

public static void main(String[] args) {

Task43subclass obj = new Task43subclass();

obj.method();

}

@Override

public int method() {

int volume = length\*breadth\*height;

System.*out*.println("Volume is "+ volume);

return volume;

}

}

44. Write code for Interface and create class to implement that interface

package concepts;

public interface Task44interface {

int *length* = 10;

int *breadth* = 10;

int *height* = 10;

public abstract int method();

}

package concepts;

public class Task44class implements Task44interface {

public static void main(String[] args) {

Task44class obj = new Task44class();

obj.method();

}

@Override

public int method() {

int volume = *length*\**breadth*\**height*;

System.*out*.println("Volume is "+ volume);

return volume;

}

}

45. Write code to add items to Hash map

**package** concepts;

**import** java.util.HashMap;

**public** **class** Task53 {

@SuppressWarnings({ "unchecked", "rawtypes" })

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

HashMap map = **new** HashMap();

map.put(1, "Java");

map.put(2, "Hadoop");

map.put(3, "SQL");

System.***out***.println("Map values: "+ map);

}

}

46. Write code to add items to Array List Collection

package concepts;

import java.util.\* ;

public class Task46 {

public static void main (String[] args)

{

ArrayList<String> Task46 = new ArrayList<String>();

Task46.add("Java");

Task46.add("Selenium");

Task46.add("QTP");

Task46.add("QC");

Task46.add("UFT");

System.*out*.println("List of elements:");

for (int i=0; i<Task46.size(); i++)

System.*out*.println(Task46.get(i) );

System.*out*.println("Size of array: " + Task46.size() );

}

}

47. Write code to add items to Hash set

**package** concepts;

**import** java.util.HashSet;

**public** **class** Task54 {

@SuppressWarnings({ "rawtypes", "unchecked" })

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

HashSet set = **new** HashSet();

set.add(1);

set.add(2);

set.add(3);

set.add(3);

set.add(4);

System.***out***.println("Size of Hash set: "+ set.size());

}

}

48. Write code to add items to integer, String array

49. Write code to connect to JDBC to get rows from employee table.

String selectTableSQL = "Query";

Statement statement = dbConnection.createStatement();

ResultSet rs = statement.executeQuery(selectTableSQL);

while (rs.next()) {

String name = rs.getString("name");

}

50. Write code to handle exceptions using try/catch/finally

package concepts;

public class Exceptions {

public static void main(String args[]) {

try {

int a[] = new int[2];

System.*out*.println("Access element three :" + a[3]);

}

catch(ArrayIndexOutOfBoundsException e) {

System.*out*.println("Exception1: " + "Out of bound");

}

try{

int a = 10, b = 0;

int c = a / b;

System.*out*.println ("Result= " + c);

}

catch(ArithmeticException e){

System.*out*.println ("Exception2: " + "Arithmetic Exception");

}

}

}

51. Write code to retrieve items from array list (for loop)

package concepts;

import java.util.\* ;

public class Task46 {

public static void main (String[] args)

{

ArrayList<String> Task46 = new ArrayList<String>();

Task46.add("Java");

Task46.add("Selenium");

Task46.add("QTP");

Task46.add("QC");

Task46.add("UFT");

System.*out*.println("List of elements:");

for (int i=0; i<Task46.size(); i++)

System.*out*.println(Task46.get(i) );

System.*out*.println("Size of array: " + Task46.size() );

String thirdElement = Task46.get(2);

String fifthElement = Task46.get(4);

System.*out*.println("Third element is: " + thirdElement);

System.*out*.println("Fifth element is: " + fifthElement);

}

}

52. Write code to retrieve items from integer, String array

53. Write code to retrieve items from Hash map

**package** concepts;

**import** java.util.HashMap;

**public** **class** Task53 {

@SuppressWarnings({ "rawtypes", "unchecked" })

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

HashMap map = **new** HashMap();

map.put(1, "Java");

map.put(2, "Hadoop");

map.put(3, "SQL");

System.***out***.println("Map values: "+ map.get(3));

System.***out***.println("Map values: "+ map.get(2));

System.***out***.println("Map values: "+ map.get(1));

}

}

54. Write code to retrieve items from Hash set

55. Write method to return list of rows code to loop throughs

package concepts;

import java.util.\* ;

public class Task55 {

public static void main (String[] args)

{

ArrayList<String> Task55 = new ArrayList<String>();

Task55.add("Java");

Task55.add("Selenium");

Task55.add("QTP");

Task55.add("QC");

Task55.add("UFT");

int n = Task55.size();

System.*out*.println("Elements are: ");

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

String n1 = Task55.get(i);

System.*out*.println(""+n1);

}

}

}