CORE -JAVA

**Core Java Demo class 1:**

-3- Topics

1. **class-keyword**: it is a factory which generates objects.

public class Ex1 {

}

1. **new-Keyword**: it will tell class to create to an object and once the object is created, new keyword gets the address of the object and will be stored in reference variable.

public class Ex2 {

public static void main (String[] args){

Ex2 ex= new Ex2(); // object is created here.

}

}

1. **garbage-collection**: - it helps us to manage the memory in an efficient manner, i.e., it regularly removes the unused objects from the memory, hence avoids memory overflow.

public class Ex3 {

//refer notes

}

**Core Java Demo class 2:**

-4- Topics

1. **Static Variables and Non Static Variables:**

public class Ex1 {

int x = 20;// non static stores only in object without object we can't asses it.

static int y = 50;// static member stores only in class common memory and we call it by using class name. variable name.

public static void main(String[] args) {

Ex1 ex = new Ex1();// object creation

System.out.println(Ex1.y);//reading static member

System.out.println(ex.x);//reading non-static member

}

}

1. **Example:**

public class Ex2 {

int x = 10;// non static main copy.

public static void main(String[] args) {

Ex2 ex = new Ex2();// object 1 1st x copy and change value but it will not effect the main copy.

ex.x = 30;// changing copy of x value.

Ex2 ex1 = new Ex2();// object 2 2nd x copy it will be same as 10 it wont be 30.

System.out.println(ex.x);// printing value of 1st obj value which we changed.

System.out.println(ex1.x);// printing value of 2nd obj copy.

}

}

1. **Example:**

public class Ex2 {

int x = 10;// non static main copy.

public static void main(String[] args) {

Ex2 ex = new Ex2();// object 1 1st x copy and change value but it will not effect the main copy.

ex.x = 30;// changing copy of x value.

Ex2 ex1 = new Ex2();// object 2 2nd x copy it will be same as 10 it wont be 30.

System.out.println(ex.x);// printing value of 1st obj value which we changed.

System.out.println(ex1.x);// printing value of 2nd obj copy.

}

}

1. **Heap & Stack:**

* Heap: In java every Objects which gets created is stored in heap.
* Stack: It helps us to maintain the programs execution flow.

**Core Java Demo class 3:**

-4- Programs

1. **Example:**

//Practice with stack and heap concept

//Methods(basics):

public class Ex1 {

public static void main(String[] args) {

Ex1 ex = new Ex1();

ex.test();

}

public void test() {//non static methods

System.out.println(100);

}

}

//Stack: program execution flow is maintained in stack.

//Heap: all the objects are created in heap memory.

1. **Example:**

// write stack and heap model to understand "LIFO"

public class Ex2 {

public static void main(String[] args) {// step 1

System.out.println(10);// step 2

Ex2 ex = new Ex2();// step 3

ex.test();// step 4//step 9

}// step 10

public void test() {// step 5// non static methods

//it will always be stored in object.

Ex2 ex1 = new Ex2();// step 6

System.out.println(ex1);// step 7

}// step 8

}

1. **Example:**

//write stack and heap model to understand "LIFO"

public class Ex3 {

public static void main(String[] args) {//step 1

Ex3.test();//calling by class name so it means it is static method

//step 2//step 6

}//step 7 stop

public static void test() {//static method//step 3

System.out.println(100);//step 4

}//step 5

}

1. **Example:**

//write stack and heap model to understand "LIFO"

public class Ex4 {

public static void main(String[] args) {// step 1

Ex4 ex = new Ex4();// step 2

ex.test1();// step 3//step 11

}// step 12 stop

public void test1() {// step 4//non static method

Ex4.test2();// step 5//step 9

}// step 10

public static void test2() {// step 6//static method as methoed is

// called with class name

System.out.println(100);// step 7

}// step 8

}

**Core Java Demo class 4:**

-1- Installation

1. Eclipse installation and shortcuts.

**Core Java class 5:**

-3- Topics, 11- Programs

1. **LOCAL VARIABLES:**
   1. Example:

//\*Local Variable: it should be created inside a method and can be used only within created method , outside created method these variables are not accessible hence below program throws an error.

public class Ex1 {

public static void main(String[] args) {

int age =30;

System.out.println(age);

Ex1 ex = new Ex1();

ex.test1();

}

public void test1() {

//System.out.println(age);//error

}

}

* 1. Example:

//correct way of writing program

public class Ex2 {

public static void main(String[] args) {

int age =30;

System.out.println(age);//only can be access within the method itself.

}

}

* 1. Example:

// how we can access local variables.

public class Ex3 {

public static void main(String[] args) {

Ex3 ex3 = new Ex3();

ex3.test1();

// System.out.println(age); we cannot access here since the variable is created in test1()//method.//error

}

public void test1() {

int age =30;

System.out.println(age);//we can access here because age is created inside test1 method.

}

}

* 1. Example:

//Without inistialising local variable , if used then it iwll give an error

public class Ex4 {

public static void main(String[] args) {

@SuppressWarnings("unused")

int weight;

//System.out.println(weight);//error//we have to initialize local variable.// even we cannot store null value in it or assign null value to it, it will give an error.

}

}

* 1. Example:

// while writing big numbers if it is difficult to read such as example (5000000) we can't put "," here so instead we can put "\_" here to not give error as shown in bellow example.

public class Ex5 {

public static void main(String[] args) {

// int salary=50,00,000;// we can't put commas.

int salary = 50\_00\_000;// we can write it this way to read it easily.

System.out.println(salary);

}

}

1. **Static Variables:**
   1. Example:

/\*Static Variables:\*These variables are created inside a class but outside method.

\*We use static keyword to create these variable.

\*It can be access anywhere in the program.

\*It is similar to Global Variable;

\*It is not Mandatory to initialize static variable, then it prints output Based on data type of static variable it will stored,for example: int-0, float-0.0, string-null. as shown in "Ex9.java"

\*\*/

public class Ex6 {

static int age = 30;// static variable so that it can be access anywhere in the program. It is

// similar to Global Variable.

public static void main(String[] args) {// main method

System.out.println(Ex6.age);

Ex6.test();// calling static method.

}

public static void test() {// static method

System.out.println(Ex6.age);

Ex6 ex = new Ex6();

ex.test1();// calling non static method.

}

public void test1() {// non static method.

System.out.println(Ex6.age);// static members can be access anywhere.

}

}

* 1. Example:

// There are three ways we can accept static variables, as shown in below program.

public class Ex7 {

static int weight = 40;

@SuppressWarnings("static-access")

public static void main(String[] args) {

System.out.println(Ex7.weight);// accepted to access static variables.

System.out.println(weight);// even this is accepted to access static variables.

//never do this//

Ex7 ex = new Ex7();

System.out.println(ex.weight);// it will give warning but no error , compiler will convert this ex.weight ->

// Ex7.weight// this is a very bad practice to access static variables.

}

}

* 1. Example:

/\*

\* static variable name and local variable names can be same but when we should know the imp thing which is shown below. \*/

public class Ex8 {

static int age = 30;// static variable

public static void main(String[] args) {

int age = 50; // even local variable name can be same as static variable.

System.out.println(age);// IMPORTANAT only local variable is given preference if we access like this

// in case if we need to access the static variable in particular we have to do

// the below thing shown

System.out.println(Ex8.age);//accessing static variable in particular, when local variable has same name as that of the static variable.

}

}

* 1. Example:

// refer Ex6 for theory.

public class Ex9 {

static int age;// Integer DT ->0

static float weight;// float DT ->0.0

static String name;// String Data Type -> null

static char ch;// Character DT -> empty space

static boolean bool; // Boolean DT -> false

public static void main(String[] args) {

System.out.println(Ex9.age);

System.out.println(Ex9.weight);

System.out.println(Ex9.name);

System.out.println(Ex9.ch);

System.out.println(Ex9.bool);

}

}

* 1. Example:

/\*Conventions:

\* 1) Blue color variable-> non static (non-italic)

\* 2) Blue color variable-> static ( italic)

\* 3) Brown color variable-> local Variable.

\*

\* 4) Class -> always write the starting letter with upper-case letter followed lower-case, when any second word it should again start with upper-case without any spaces followed by lower-case [Bank, BankAccount].

\*

\* 5) variable -> always starts with lower-case and continue same, if two words 2nd word should be written in upper-case.[age,yourAge,thisIsHerName]--> special character is allowed is only [$,number,\_] -->we can write "$ or \_" anywhere beginning middle or at end or we can also just name it $ and \_ it will not give error,IMPORTANT thing is variable name can never begin with number but it can be in middle or at the end.

\*

\* 6) Method -> same like variables with "()" as suffix.

\*

\* 7) Keywords -> all the keywords starts with lower-case

\*/

public class Ex10 {

}

1. **Non-Static Variables:**

/\*non-static variable:1)non static variables are created outside methods and inside class without static keyword.

\* 2) These variables can never be accessed without creating object

\* 3) It is not mandatory to initialize non static variable , because depending on data-type default value gets stored in it, They are also called as instance variable.

\*/

public class Ex11 {

int age1;// Integer DT ->0

float weight1;// float DT ->0.0

String name1;// String Data Type -> null

char ch1;// Character DT -> empty space

boolean bool1; // Boolean DT -> false

public static void main(String[] args) {

Ex11 ex = new Ex11();

System.out.println(ex.weight1);

System.out.println(ex.age1);

System.out.println(ex.name1);

System.out.println(ex.ch1);

System.out.println(ex.bool1);

}

}

**Core Java class 6:**

-4- Topics

1. **Data Types:**

//DataTypes-->memory size --> Default Value

public class Ex1 {

static byte b; //-> 1 Byte -> 0

static short s; //-> 2 Byte -> 0

static int i; //-> 4 Byte -> 0

static float f; //-> 4 Byte -> 0.0

static long l; //-> 8 Byte -> 0

static double d; //-> 8 Byte -> 0.0

static char c; //-> 2 Byte -> empty space

static boolean bool; //-> N/A -> false

static String str; //-> N/A -> null

public static void main(String[] args) {

System.out.println(b);

System.out.println(s);

System.out.println(i);

System.out.println(l);

System.out.println(f);

System.out.println(d);

System.out.println(c);

System.out.println(bool);

System.out.println(str);

}

}

1. Example:

//long: when we exceed the integer range we terminate with l or else l as suffix is not needed

//float: terminated with f

//char: should always be stored in ''.

public class Ex2 {

public static void main(String[] args) {

String s= "hello";

char c='a';

int a= 20;

long l = 6361317502l;// if it exceeds the int range then only we have to end it with l.

long l2 =10;

float f=20.54f;

double d=287.98;

boolean b = true;

System.out.println(s);

System.out.println(c);

System.out.println(a);

System.out.println(l);

System.out.println(l2);

System.out.println(f);

System.out.println(d);

System.out.println(b);

}

}

1. **Var -Type:**

// var-type:1) this was introduced in version 10 of java,

/\* 2)var-type can store any kind of value in it as shown in below example

\*

\* 3) var-type cannot be static and non static variable, it can be only local variable.

\* 4) var-type cannot be method argument,hence the below program throws error.

\*

\* 5) var can have a variable name as a var itslf as it is atype not an data type.

\*

\* 6) "-------"-------"

\*

\* 7) even when we write any keyword and its name is String and if we print output also it will not give error.

\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*2)

\* public class Ex3{

\* psvm(){

\* var x1 = "pankaj";

\* var x3 =10;

\* var x7 =30.98;

\* var x8 = 'a';

\* var a1= new Ex3();

\* sop(x1);

\* sop(x3);

\* sop(x7);

\* sop(x8);

\* sop(a1);

\* }

\* }

\* \*/

/\*3)

\* public class Ex3{

\* static var x2 =10;

\* var x3 =20;

\* psvm(){

\* var x1="pankaj"

\* }

\* }

\* \*/

/\*4)

\* public class Ex3{

\* psvm(){

\* Ex3 ex = new Ex3();

\* ex.test(10);

\* }

\* p v test(var x){// error

\* sop(x)

\* }

\* \*/

/\*5)

\* class Ex3 {

\* psvm(){

\* var var = 10;//it wont give error where as if it is a keyword it wont accept this syntax so we consider var as not a keyword.

//int int =20;//error

\* sop(var);

\* }

\* }

\* \*/

/\*6)

\* class Ex3 {

\* psvm(){

\* var String = 10;

\* sop(String)

\* }

\* }

\* \*/

/\*7)

\* class Ex3{

\* psvm(){

\* int String = 30;

\* sop(String); //no error it will print special case

\* }

\* }

\* \*/

public class Ex3 {

}

1. **Reference Variable:**

/\*reference variable : 1) can store objects address or null value

\* 2) if a reference variable is created in a method then it is a local variable.

\* 3) if you make reference variable static it has global access.

\* 4) Data type of reference variable is class name.

\* \*/

public class Ex4 {

static Ex4 ex;

public static void main(String[] args) {

Ex4 ex = new Ex4();//

System.out.println(ex);

ex.test();

}

public void test() {

System.out.println(ex);

}

}

**Core Java class 7:**

-2- Topics, 11- Programs

1. Methods:

//more on methods

public class Ex1 {

public static void main(String[] args) {

Ex1 ex = new Ex1();

ex.test(10, 10.3f, "test", true, 'a');

}

public void test(int i, float f, String string, boolean b, char c) {

System.out.println(i);

System.out.println(f);

System.out.println(string);

System.out.println(b);

System.out.println(c);

}

}

1. Example:

//special way to pass arguments into method.

public class Ex2 {

public static void main(String[] args) {

Ex2 ex = new Ex2();

ex.test(10, 20, 30, 40);

}

private void test(int... i) {//Array concept

System.out.println(i[0]);

System.out.println(i[1]);

System.out.println(i[2]);

System.out.println(i[3]);

}

}

1. Example:

/\* void :1)If a method is void then it means it cannot return any value

\* hence the below program throws error.

\* 2) we cannot anything after return keyword.

\* 3) we cannot write anything side to return keyword in void method ,

\* we can just put return keyword terminated with semicolon.

\* 4) we cannot just write return keyword without any value next to it in non void method.

\* \*/

public class Ex3 {

public static void main(String[] args) {

Ex3 ex = new Ex3();

ex.test();

int test2 = ex.test2();// assigning non void member's to local variable.

System.out.println(test2);// printing non void values

}

private int test2() {

return 100;// it will be thrown out and we should assign the call statement to variable.

// System.out.println(20);//error we can't write anything after return

}

private void test() {

// return 100;//error

System.out.println(100);// we can write anything before return keyword but not after.

return;// accepted

// System.out.println(20);//error we can't write anything after return

}

}

1. Example:

public class Ex4 {

public static void main(String[] args) {

Ex4 ex = new Ex4();

String var = ex.test();

System.out.println(var);

}

private String test() {// non void method

return "pankaj";

}

}

1. Example:

// if you write anything after return keyword then that will not run and it will give us unreachable code error, as shown in the below example.

public class Ex5 {

public static void main(String[] args) {

Ex5 ex = new Ex5();

String var = ex.test();// getting value from non void method and stores in local variable for printing.

System.out.println(var);// printing Stored value.

}

private String test() {// non void method

System.out.println(100);// it will run it will not be an unreachable code as it is before return

// keyword.

return "pankaj";// we have to write value after return as it's an non void method or else error.

// System.out.println(100);// error unreachable code

}

}

1. Example:

/\* A)return :

\* 1) return keyword can be used only in void method.

\* 2) its optional.

\* 3) It returns control to method calling statement.

\* 4) We can write return keyword in main method too.

\* 5) We can write return in static method too.

\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* B) return value:

\* 1) cannot be used in void methods.

\* 2) returns control and value to the calling statement.

\* 3) if not a void method its mandatory to use [return value]

\* \*/

public class Ex6 {// A

public static void main(String[] args) {

Ex6 ex = new Ex6();

ex.test();

return;// it wont give any error

}

private void test() {

return;

}

}

/\*

\* public class Ex6 {//B public static void main(String[] args) { Ex6 ex = new

\* Ex6(); int a=ex.test(); System.out.println(a) return;// it wont give any

\* error }

\* private int test() { return 100; } }

\*/

1. Example:

// Continuation of 6 just some examples for methods.

public class Ex7 {

public static void main(String[] args) {

Ex7 ex = new Ex7();

ex.test();

return;// it wont give any error

}

private void test() {

System.out.println(100);

return;

// System.out.println(100);// error

}

}

1. Constructors:

/\*CONSTRUCTOR:

\* 1) It should have same name as that of Class

\* 2) Every time we create an object a constructor is called.

\*

\*

\* \*/

public class Ex8 {

// constructor start

Ex8() {// constructor should have the same name as that of the class its mandatory.

System.out.println(100);

}

// constructor end

// main method start

public static void main(String[] args) {

new Ex8();// 1st time, every time we create an object constructor is called.

new Ex8();// 2nd time object created so 2 times constructor is called.

}

// main method end

}

1. Example:

//constructor with arguments.

public class Ex9 {

Ex9(int x) {// constructor with argument

System.out.println(x);

}

public static void main(String[] args) {

new Ex9(10);// object created.

}

}

1. Example :

//fake constructor / void fakeism.

public class Ex10 {

void Ex10() {// this is not a constructor its an nonstatic method.

System.out.println(100);

}

public static void main(String[] args) {

new Ex10();// no output as it is not an constructor.

}

}

1. Example:

// continuation of Ex10

// Note: method name and class name can be same and wont give error until it has same arguments if so it will give error, as shown below.

public class Ex11 {

void Ex11() {// method not constructor.

System.out.println(100);

}

public static void main(String[] args) {

Ex11 ex = new Ex11();// object created but it wont call constructor as the above thing is void method

// so never call the constructor with void as prefix.

ex.Ex11();// we write this to access the non-static method.

}

}

**Core Java class 8:**

-2- Topics, 4- Programs

1. Difference Between JRE and JDK:

/\*

\* Difference between jre and jdk:

\*

\* A)jre:java runtime environment.

\* 1)it is used to run the java projects or files.

\* 2)if you want to run only .class files then install jre, customers use this to run programs

\*

\* B)jdk: java development kit

\* 1)it is used to compile and run the program.

\* 2)Developers use this to develop projects.

\*

\* \*/

/\*ASCII only one language but to write in multilanguage we have UNICODE.

\* Note: bin -> all the .class files are present here.

\* Note: src -> all your java files are present.

\* \*/

//uni-code example.

public class Ex1 {

public static void main(String[] args) {

int x = 'a';// English language

System.out.println(x);

int y = 'ए';// Hindi language.

System.out.println(y);

}

}

1. Constructor continuation:

//more on constructor

//

// new keyword will request to class to create an object.

// new keyword will call constructor that is mandatory.

// new keyword will gets object address and stores that in reference variable.

// if we create an object without an argument and explicitly constructor is not created by the programmer then during compilation in .class file empty body constructor gets added as shown in the below example.

public class Ex2 {

/\*Ex2(){

\*\*empty body constructor is created by compiler in .class file automatically\*\*

\* }

\* \*/

public static void main(String[] args) {

new Ex2();// object created an empty body constructor is created by new keyword automatically.

}

}

1. Example:

// if we create an object with a value, if constructor is not created it will give error, creating constructor here is mandatory.

public class Ex3 {

Ex3(int i) {// mandatory creation of constructor with value or else it will give error.

System.out.println(i);

}

public static void main(String[] args) {

new Ex3(100);// object with value// if constructor is not created then it will give error.

}

}

1. Example:

// creating constructor with value is mandatory if we pass value in object or else it will give error.

public class Ex4 {

public static void main(String[] args) {

//Ex4 ex = new Ex4(20);// error

}

}

**Core Java class 9:**

-2- Topics, 8- Programs

1. Constructor overloading

* Example1:

package p1;

/\*Constructor overloading:

\* 1) here we create more than one constructor in same class.

\* 2) create constructors with different number of arguments or different type of argument

\*

\* \*/

public class Ex1 {

Ex1() {// no of arguments=0.

System.out.println(300);

}

Ex1(int i) {// no of arguments = 1

System.out.println(i);

}

Ex1(int i, int y) {// no of arguments = 2

System.out.println(i);

System.out.println(y);

}

@SuppressWarnings("unused")

public static void main(String[] args) {

Ex1 ex = new Ex1();// object creation 1

Ex1 ex1 = new Ex1(10);// object creation 2

Ex1 ex2 = new Ex1(100, 200);// object creation 3

}

}

* Example 2:

package p1;

/\*Constructor overloading.

\* 1)different type of argument

\* \*/

public class Ex2 {

Ex2(int i) {// no of arguments = 1 type = int

System.out.println(i);

}

Ex2(char i) {// no of arguments = 1 type = char

System.out.println(i);

}

Ex2(String i) {// no of arguments = 1 type = string

System.out.println(i);

}

public static void main(String[] args) {

new Ex2(10);// object creation 1

new Ex2('a');// object creation 2

new Ex2("pankaj");// object creation 3

}

}

* Example 3:

package p1;

/\*Constructor overloading

\* 3) if those 2 different arguments belong to same data type like int and byte etc.

\* Note: in the below program, it will call int because all numbers by default is treated as int.

\* \*/

public class Ex3 {

Ex3(int i) { // no of arguments = 1 type = int

System.out.println(i);

System.out.println("int");

}

Ex3(byte i) {// no of arguments = 1 type = byte

System.out.println(i);

System.out.println("byte");

}

public static void main(String[] args) {

new Ex3(10);// object creation 1

// \*\*\* to pass byte value we write as new //Ex3((byte)10);

// \*\*\* to pass long value we write as new //Ex3(10L);

// in java if we write any number it will always be treated as int, if we

// explicitly want to pass byte value then we do down casting.

}

}

* Example 4:

package p1;

/\* Note: class name constructor name method name and variable name can be same as shown in the below example.

\* \*/

public class main {

main() {

System.out.println(1000);

}

public static void main(String[] args) {

int main = 10;

main m = new main();

System.out.println(main);

}

}

1. Packages:

* Example 5:

package p1;

/\* Package:

\* 1) packages are nothing but folder, that is to store all data organized way.

\* 2) package names cannot be keywords or java or upper-case.

\* 3) we cannot write anything above package it should be the first word in program

\* 4) packages are folders created in java to store programs in organised manner.

\* \*/

public class Ex5 {

}

* Example 6:

package p1;

import p2.Ex1p2; // we have to write it mandatorily.

import p3.p4.p5.A; // we have to write it mandatorily

/\*CONTINUATION OF Ex5

\*continuation from package p2->Ex1p2-> import p2.Ex1p2;.

\* continuation from package p3.p4.p5->A-> import p3.p4.p5.A;.

\* \*/

public class Ex6 {

@SuppressWarnings("unused")

public static void main(String[] args) {

Ex5 ex = new Ex5(); // object of Ex5 from same package p1, accessing in different class.

Ex1p2 ex1 = new Ex1p2(); // object of Ex1p2 from different package p2, accessing in different class of

// different package.

A a = new A(); // object of A from different package p3.p4.p5, accessing in different class of

// different package.

}

}

* Example 7:

package p2;

public class Ex1p2 {

}

* Example 8:

package p3.p4.p5;

public class A {

}

**Core Java class 10:**

-2- Topics

1. Packages continuation:

package p1;

public class A {

}

package p2;

/\* Packages:

\* 1) helps us to resolve naming convention problem\* \*/

public class B {

public static void main(String[] args) {

p1.A a1 = new p1.A();// 2nd method to import other packages. this is use-full when 2 packages have class of same name we use this type of accessing (packagename.classname)

p3.A a2= new p3.A();

}

}

package p3;

public class A {

}

1. OOPS CONCEPT- inheritance:

* Example:

package inheritance;

/\* Inheritance:

\* 1) here we inherit the members of parent class to child class with an intention of reusability.

\*

\* \*/

public class Ex1 {// parent class

int x = 10;

}

package inheritance;

// NOTE: only one class should have public when they are created in same file, and which is public should be name of class, interview question.

/\*Extends: is a keyword which helps us to inherit\*/

public class Ex1\_continuation extends Ex1 {// child class //replicating which are present in parent class to child name.

// it is not moved they are copied.

public static void main(String[] args) {

Ex1\_continuation ex = new Ex1\_continuation();// object stores all the non-static variables from both parent and

// class name

System.out.println(ex.x);

}

}

* Example:

package inheritance;

public class Ex2 {

int x= 10;

public void test() {

System.out.println(100);

}

}

package inheritance;

public class Ex2\_continuation extends Ex2 {

public static void main(String[] args) {

Ex2\_continuation ex = new Ex2\_continuation(); //here method and variables gets stored from parent class

ex.test();

System.err.println(ex.x);

}

}

* Example:

package inheritance;

// in the below example A and B are non-subclasses and there is no inheritance happening between them

public class Ex3 {

int x = 10;

}

package inheritance;

// no inheritance

public class Ex3\_c {

public static void main(String[] args) {

Ex3 ex = new Ex3();

System.out.println(ex.x);

}

}

* Example:

package inheritance;

// we cannot do multiple inheritance in java.

public class Ex4 {

public void test1() {

System.out.println(100);

}

}

package inheritance;

// continuation of Ex4

public class Ex4\_1 extends Ex4{

public void test2() { // test1() inherited

System.out.println(200);

}

}

package inheritance;

//object-oriented programming is applicable only in non-static.

public class Ex4\_2 extends Ex4\_1 {

public void test3() {// test1() test2() inherited

System.out.println(300);

}

public static void main(String[] args) {

Ex4\_2 ex = new Ex4\_2();

ex.test1();

ex.test2();

ex.test3();

}

}

* Example:

package inheritance;

//multiple inheritance in java is not possible/supported in class level, as it is shown below.

public class Ex5 {

}

package inheritance;

public class Ex5\_1 {

}

package inheritance;

public class Ex5\_2 /\* extends Ex5 , EX5\_1\*/{

// we cannot inherit multiple classes.

}

* Example:

/\* we cannot inheritance without importing package where other program is present in other class and other package. \*/

package inheritance;

public class Ex6 {

int x = 10;

}

package inheritance\_1;

import inheritance.Ex6;//importing package is mandatory.

public class Ex6\_1 extends Ex6{

public static void main(String[] args) {

Ex6\_1 ex = new Ex6\_1();

System.out.println(ex.x);

}

}

1. OOPS CONCEPT- polymorphism:

* Example:

package polymorphism;

/\* polymorphism

\* 1) it can only be applied on methods and not variables.

\* 2) here we develop a feature such that it can take more than one form.

\* \*/

/\*OVERLOADING

\* 1) inheritance is mandatory to do this overloading.

\* 2) here we inherit a method from parent class and then we modify the logic of inherited method in child class by once again creating a method with a same signature in the child class

\* \*/

public class Ex1 {

public void test() {

System.out.println(100);

}

}

package polymorphism;

//overloading

public class Ex1\_1 extends Ex1 {

public void test() { // here overloading is happening. here the copy is not taking place, here test method is taking 2 forms that is called as polymorphism

System.out.println(200);

}

public static void main(String[] args) {

Ex1\_1 ex = new Ex1\_1();

ex.test();

}

}

* Example:

package polymorphism;

/\*overloading second example.\*/

public class Ex2 {

public void test() {

System.out.println(100);

}

}

package polymorphism;

/\*overloading happens here the same method from class Ex2 is getting logic modified. \*/

public class Ex2\_2 extends Ex2 {

public void test() {// overloading happens here the same method from class Ex2 is getting logic

// modified.

System.out.println(500);

}

public void test1() {

System.out.println(100);

}

public static void main(String[] args) {

Ex2\_2 ex = new Ex2\_2();

ex.test();

ex.test1();

}

}