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);

}

}