**Constructors:-**

**Whenever we are creating an object some piece of the code will be executed automatically to perform initialization state of an object this piece of the code is known as constructor.**

**Example:-**

**class Test**

**{**

**Test()//constructor name and class name both are same.**

**{**

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

**}**

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

**{**

**Test t=new Test();**

**}**

**}**

**Result:- constructor invoked**

**There are two types of constructor:-**

1. **Default Constructor.**
2. **User Defined Constructor**

**Default constructor:-**

* **If our class doesn’t contain any constructor then java compiler provides default constructor.**
* **Inside the class default constructor is invisible mode.**
* **It is always no parameter constructor.**

**This application not contains any constructor.**

class Test

{

void m1()

{

System.out.println("m1()method");

}

public static void main(String[]args)

{

Test t=new Test();

t.m1();

}

}

**After compilation java compiler provide default constructor**.

class Test

{

void m1()

{

System.out.println("m1() method");

}

**Test()//default constuctor**

**{**

**}**

public static void main(String[]args)

{

Test t=new Test();

t.m1();

}

}

**After**

**Compilation**

**User Defined Constructor: -**

**The constructor which is defined by user (programmer) is called user defined constructor.**

**User Defined Constructor**

1. **Non-parameterized Constructor**
2. **Parameterized Constructor**

**Non-Parameterized Constructor:-**

**The user defined constructor which are not contain any parameter such types of constructor are called Non-parameterized constructor.**

**Example:-**

**class Test**

**{**

**int x;**

**int y;**

**Test()**

**{**

**x=10;**

**y=20;**

**}**

**public void m1()**

**{**

**System.out.println(x+".."+y);**

**}**

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

**{**

**Test t1=new Test();**

**Test t2=new Test();**

**t1.m1();**

**t2.m1();**

**}**

**}**

**Result:-**

**10..20**

**10..20**

**Parameterized Constructor:-**

**The user defined constructor which are contain at least one parameter such types of constructors are called parameterized constructor.**

**In case Default constructor and non-parameterized constructor for every object we get same or default value of all instance variable but in case non-parameterized constructor, for every object we get values of instance variable, based on parameter.**

**Example:-**

**class Test**

**{**

**Test(int a)**

**{**

**System.out.println("1-arg parameterized constructor");**

**}**

**Test(int a,int b)**

**{**

**System.out.println("2-arg parameterized constructor");**

**}**

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

**{**

**Test t1=new Test(10);**

**Test t2=new Test(10,20);**

**}**

**}**

**Result:-**

**1-arg parameterized constructor**

**2-arg parameterized constructor**

**Copy Constructor:- By using constructor if we want to copy the value of one object to another object called copy constructor.**

**Example:-**

**class Test**

**{**

**int x;**

**int y;**

**Test(int x,int y)**

**{**

**this.x=x;**

**this.y=y;**

**}**

**Test(Test t)**

**{**

**this.x=t.x;**

**this.y=t.y;**

**}**

**void show()**

**{**

**System.out.println(x+".."+y);**

**}**

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

**{**

**Test t2=new Test(10,20);**

**Test t3=new Test(t2);**

**t2.show();**

**t3.show();**

**}**

**}**

**Result:-**

**10..20**

**10..20**

**Constructor overloading:-Inside a class if we are declaring multiple constructor with different parameter such types of constructor are called overloaded constructor and in java constructor overloading is possible.**

**Example:-**

**class Test**

**{**

**Test()**

**{**

**System.out.println("No-arg constructor");**

**}**

**Test(int a)**

**{**

**System.out.println("int arg constructor");**

**}**

**Test(double d)**

**{**

**System.out.println("double arg constructor");**

**}**

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

**{**

**Test t1=new Test();**

**Test t2=new Test(10);**

**Test t3=new Test(12.22);**

**Test t4=new Test(10l);**

**}**

**}**

**Result:-**

**No-arg constructor**

**int arg constructor**

**double arg constructor**

**double arg constructor**

**Note:-In java constructor overloading concept is applicable, but constructor overriding concept is not applicable.**

**Example:-**

**class Parent**

**{**

**Parent()**

**{**

**System.out.println("Hello parent");**

**}**

**}**

**class Child extends Parent**

**{**

**Child(int a)**

**{**

**super();**

**System.out.println("Hello Child");**

**}**

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

**{**

**//Child c1=new Child();**

**Child c2=new Child(10);**

**}**

**}**

**Result:-**

**Hello parent**

**Hello Child**

**Note:-Every java class including abstract class can contain constructor but interface can’t contain constructor.**

**Case Study:-**

**Case1:-**

**Inside a class if we declare at least one constructor it may be parameterized or non-parameterized then java compile won’t create default constructor.**

**Example:-**

**class Test**

**{**

**Test(int a)**

**{**

**System.out.println("1-arg user defined constructor");**

**}**

**Test(int a,int b)**

**{**

**System.out.println("2-arg user defined constructor");**

**}**

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

**{**

**Test t1=new Test();**

**Test t2=new Test(10);**

**Test t3=new Test(10,20);**

**}**

**}**

**Result: -In the above application we will get compile time error because java compiler is unable to create default constructor, java compiler is created default constructor if and only if there is no any user defined constructor is available.**

**Compile Time Error is:-**

**Test.java:13: error: no suitable constructor found for Test(no arguments)**

**Test t1=new Test();**

**Case2:-**

**Assign the value of instance variable inside constructor and method [Constructor vs method] then method assigns instance variable we be invoked because method will be executed last then constructor.**

**Example:-**

**class Test**

**{**

**int x;**

**int y;**

**Test(int x,int y)**

**{**

**this.x=x;**

**this.y=y;**

**}**

**void m1(int x,int y)**

**{**

**this.x=x;**

**this.y=y;**

**}**

**void info()**

**{**

**System.out.println(x+".."+y);**

**}**

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

**{**

**Test t=new Test(10,20);**

**t.m1(100,200);**

**t.info();**

**}**

**}**

**Result:- 100..200**

**Case3:-**

**Inside a constructor if we are using super() or this() statement then this statement should be the first statement otherwise we will get compile time error.**

**Example:-**

**class Test**

**{**

**Test()**

**{**

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

**super();**

**}**

**}**

**Result:- error: call to super must be first statement in constructor**

**super();**

**Case4:-Inside the constructor we can use either super() or this() statement but not both statement simultaneously otherwise we will get compile time error.**

**Example:-**

**class Test**

**{**

**Test()**

**{**

**super();**

**this();**

**}**

**}**

**Result:-**

**error: call to this must be first statement in constructor**

**this();**

**Case5:-We can use super() and this() statement only inside the constructor no any other place otherwise we will get compile time error.**

**Example:-**

**class Test**

**{**

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

**{**

**super();**

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

**}**

**}**

**Result:-**

**error: call to super must be first statement in construct or super();**

**Case6:-**

**A constructor can call another constructor by using this() .**

**class Test**

**{**

**Test()**

**{**

**this(10);**

**System.out.println("0-arg Constructor");**

**}**

**Test(int x)**

**{**

**this(10,20);**

**System.out.println("1-arg Constructor");**

**}**

**Test(int x,int y)**

**{**

**System.out.println("2-arg Constructor");**

**}**

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

**{**

**Test t=new Test();**

**}**

**}**

**Result:-**

**2-arg Constructor**

**1-arg Constructor**

**0-arg Constructor**

**Note:-In this application when object is created 0-parameter constructor is call automatically and 0-parameter constructor call 1-parameter constructor and 1-parameter constructor call 2-parameter constructor by using this().**

**Case7:-**

**Recursion constructor:- constructor call each other is known as construction recursion, In our program there may be chance recursion constructor call which create compile time error.**

**Example:-**

**class Test**

**{**

**Test()**

**{**

**this(20);**

**System.out.println("0-arg constructor");**

**}**

**Test(int a)**

**{**

**this();**

**System.out.println("1-arg constructor");**

**}**

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

**{**

**Test t=new Test();**

**}**

**}**

**Result:-**

**java:8: error: recursive constructor invocation**

**Test(int a)**

**Case7:-**

**Note:-If parent class method contains any argument constructor then in child class we should provide special care with respect to constructor.**

**Note:-Whenever we writing any argument constructor inside parent class, we should also write no-arg constructor.**

**Example:-**

**//Valid**

class Parent

{

}

class Child extends Parent

{

}

**//Valid**

class Parent

{

Parent()

{ }

}

class Child extends Parent

{

}

**//Not Valid**

class Parent

{

Parent()

{ }

}

class Child extend Parent

{

}

**What is the difference between super(),this() and super, this keyword?**

|  |  |
| --- | --- |
| **super(),this()** | **Super, this keyword** |
| **1. These are constructors calls** | **1. These are keywords** |
| **2. We can use these to invoke super class &**  **current constructors directly** | **2. We can use refers parent class and current class instance members.** |
| **3. We should use only inside constructors as**  **first line, if we are using outside of**  **constructor we will get compile time error.** | **3. We can use anywhere (i.e., instance area)**  **except static area , other wise we will get**  **compile time error .** |

**Case8:-**

**If Parent class constructor throw any checked exception then child class constructor should be throws same types of exception or it Parent types otherwise we will get compile time error.**

**Example:-**

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

**class Parent**

**{**

**Parent() throws IOException**

**{**

**}**

**}**

**class Child extends Parent**

**{**

**Child()**

**{**

**}**

**}**

**Result:- unreported exception IOException; must be caught or declared to be thrown**

**Example:-To solve above problem.**

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

**class Parent**

**{**

**Parent() throws IOException**

**{**

**}**

**}**

**class Child extends Parent**

**{**

**Child()throws IOException**

**{**

**}**

**}**

**We can't create object for abstract class but abstract class can contain constructor what is the need ?**

**Abstract class constructor will be executed for every child class object creation to perform initialization of child class object only.**

**Which of the following statement is true?**

**1. Whenever we are creating child class object then automatically parent class object will be created.(false).**

**2. Whenever we are creating child class object then parent class constructor will be executed.(true)**

**Example:-**

**abstract class Parent**

**{**

**Parent()**

**{**

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

**//11394033//here this means child class object**

**}**

**}**

**class Child extends Parent**

**{**

**Child()**

**{**

**System.out.println(this.hashCode());//11394033**

**}**

**}**

**class Test**

**{**

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

**{**

**Child c=new Child();**

**System.out.println(c.hashCode());//11394033**

**}**

**}**