OVERLOADED CONSTRUCTOR

Within a class we can declare multiple constructor and all these constructors having same name but different type of argument, hence all these constructor considered as overloaded constructor. Hence overloading concept applicable for constructor.

**class** Base {

**public** Base(){}

**public** Base(**int** i) {}

**public** Base(**double** d) {}

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

**new** Base();

**new** Base(10);

**new** Base(10l);*//auto-promotion during over-loading*

**new** Base(10.5);

}

}

For constructors **inheritance and overriding** concepts are not applicable. But overloading concept is applicable

Every class in java including abstract class can contain constructor but interface cannot contain constructor

**class** Test {

**public** Test() {}

}

**abstract class** Base {

**public** Base(){}

}

**interface** inter {

**public** inter() {} //invalid because there is no chance of existing instance variable

}

This is possible

**class** Base {

**public static void** m1() {

*m2*();

}

**public static void** m2() {

*m1*();

}

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

*m1*();

}

}

Above will result in stack overflow error

Recursive constructor invocation is not possible:

**class** Base {

Base() { //Recursive constructor call is not possible

**this**(10);

}

Base(**int** i) {

**this**();

}

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

}

}

Compiler would not allow this.

**Invalid**:

**class** Base {

Base(**int** i) {}

}

**class** Child **extends** Base{

}

Will error out – cannot find Base() in class Base

**Valid:**

**1.**

**class** Base {

}

**class** Child **extends** Base{

**}**

**2.**

**class** Base {

Base() {}

}

**class** Child **extends** Base{

}

1. If parent class contains any argument constructor then while writing child classes we have to special care with respect to constructors.

2. Whenever we are writing any argument constructor it is highly recommended to write no-argument constructor also.

**Exception handling:**

**class Base {**

Base() **throws** IOException{}

}

**class** Child **extends** Base{ //Unhandled exception: IOException

}