**Constructor:**

* The main purpose is to perform initialization.

Student s1=new Student(10,”ram”);

Points:

1. Creates Student object
2. Every instance variable is created and provides a default values ( Performing initialization).
3. And create object reference variable s1.

[note: ***new*** keyword will create an object not constructor]

Rules:

1. Name should be same as class name.
2. return type not appliable.
3. Only applicable access modifiers are public, <default>, protected, private.

\*\* **Important**

[Compiler would add the default constructor if no constructor specified]

[ access modifier of the default constructor is same as class access modifier].

[default constructor contains super(), it is no args call for parent class constructor].

[ Even abstract class contains constructor ].

**super()** and **this()** are **constructor calls,**

* can use **only inside constructor** and **only one** of them but **not both** at **same time.**
* **super()** -- > refers to the super class constructor.
* **this()** -- > refers to current class constructor.

\*\* **important**:-

[note: call to super() or this() must be first line in constructor and both are not allowed at same time].

[note: call to super() or this() must be used constructor only, is not allowed in method].

**super** and **this** are used to refer **instance members** ( variables or methods).

* **super** - - >refers to super class object instance members.
* **this** -- > refers to current class object instance members.

[note: super or this can used anywhere in class it can be method or can constructor].

[note super and this can be used anywhere except static area because **static area** is not related to **object**].

**Inheritance concept- overriding:**

Inheritance concept: **Overriding** is ***not applicable*** for **constructor**.

Inheritance concept: **Overriding** is ***applicable*** for **methods**.

\* concreate class can contain constructors.

\* abstract class can contain constructors.

\* Interface cannot constructor because every variable present in interface is public static final.

Difference between

Child c = new Child();

If we know exact runtime type of object.

By using child reference we can call both parent and child class methods.

Parent p = new Child();

If we don’t know exact runtime type of object. Because parent reference object holds parent runtime type as well as child runtime type of object.

By using parent reference we can call only methods available

in parent class but cannot call child specific methods.

**Creating objects in java:**

1. using ***new*** keyword: when we know name of the class to create an object in the beginning..

**Student s1 = new Student();**

1. using newInstance() method: if we do not know the class but we only know in runtime.

**Object o**=**Class.*forName*(args[0]).newInstance()**;

**instanceof** - - > is an operator in java to check object type.

**isInstance()** - - > is a method in java to check the type of object type at runtime.