|  |  |
| --- | --- |
| 1. Constructors can be overloaded: | Yes,( Default , Prameterized) |
| 2. Default constructor: | Declaring constructor is not required. When we don’t supply any constructor, the Java compiler automatically generates a default constructor which is empty and has no parameters. |
|  | Constructor has no return type, but its return the **object reference** |
| compiler won’t generate the default constructor | The compiler won’t generate the default constructor if there’s a already constructor in the class: |
| Inheritance | Constructors are not inherited |
| Constructors can be private | Yes, this is true. We can make a **constructor private** to prevent the outside world from creating a new instance of our class. Consider the following class with a private constructor: |
| A constructor calls the default constructor of its superclass | class Parent {  Parent(int number) {  }  }    class Child extends Parent {  Child() {  }  }  Here, the code **doesn’t get compiled** because the compiler inserts a call to super() in the Child’s constructor and calls the default constructor which is not present..:  Child() {  super(); // auto-inserted by the compiler  } |
| Can you make a constructor final? | No, the constructor can't be final. Compiler error |
| differences between the constructors and methods? |  |
| a = 10 b = 15  byte gets promoted to int |  |
| o/p:  Parent default  child1 |  |
|  |  |
|  |  |
|  |  |
|  |  |