hashCode() method:

- For every object a unique number generated by **jvm** which is nothing but hashcode.
- HashCode is not the address of the object.
- jvm will use hashcode while saving object into hashing related data structures like: *hashTable*, *hashMap*, *hashSet*
- The main advantage of saving objects based on hashCode is search operation will become easy. (The most powerful search algorithm up to today is hashing)
- 1. Linear Search: O(n)
  2. Binary Search: O(log n)
  3. Hashing: O(1)

Generating hashCode:

- If you are giving the chance to Object class `hashCode()` method, it will generate hashcode based on address of the object. It doesn't mean hashcode represent address of an obejct.
- Based on our requirements we can *override* `hashCode()` method in our class to generate our own hashcode.

Overriding `hashCode()` is said to be ***proper*** if and only if for every we have to generate a unique number as a hashcode.

Improper way to override hashCode():

- generating the same hashcode for every student object.

```
class Student {
    public int hashCode() {
        return 100;
    }
}
```

proper way to override hashCode():

- generating a unique hashcode for every student object.

```
class Student {
    public int hashCode() {
        return rollno;
    }
}
```

toString() vs hashCode():

- If we giving the chance to Object class `toString()` method it will internally call `hashCode()` method.
- If we are overring `toString()` method then our `toString()` method may not call `hashCode()` method.

```
class Test {
    int i;
    Test (int i) {
        this.i = i;
    }

    public static void main(String[] args) {
        Test t1 = new Test(10);
        Test t1 = new Test(100);

        System.out.println(t1); => Test@xxxxxx
        System.out.println(t2); => Test@xxxxxx
    }
}
```

Called methods:

- Object => toString()
- Object => hashCode()

```
class Test {
    int i;
    Test (int i) {
        this.i = i;
    }
    public int hashCode() {
        return i;
    }

    public static void main(String[] args) {
        Test t1 = new Test(10);
        Test t1 = new Test(100);

        System.out.println(t1); => Test@a (10 = a in hexadecimal)
        System.out.println(t2); => Test@64 (100 = 64 in hexadecimal)
    }
}
```

Called methods:

- Object => toString()
- Test => hashCode()

```
class Test {
    int i;
    Test (int i) {
        this.i = i;
    }
    public String toString() {
        return i + "";
    }

    public int hashCode() {
        return i;
    }

    public static void main(String[] args) {
        Test t1 = new Test(10);
        Test t1 = new Test(100);

        System.out.println(t1); => 10
        System.out.println(t2); => 100
    }
}
```

Called methods:

- Test => toString()