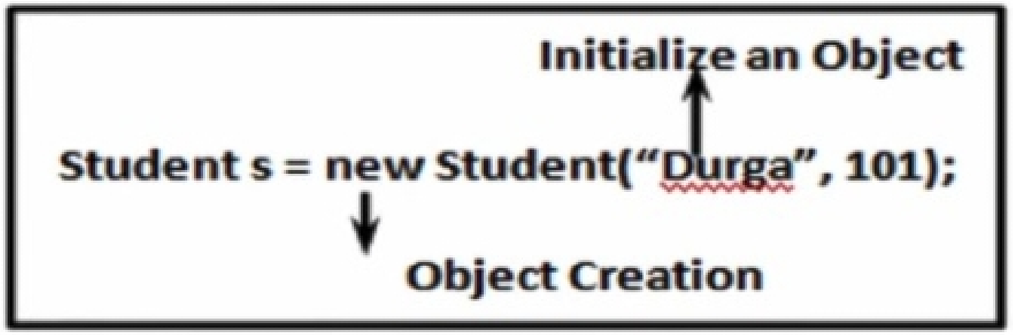
1.Roles of new keyword and Constructor

The Purpose of Constructor is

- To Initialize an Object but Not to Create an Object.
- Whenever we are creating an Object after Object Creation automatically Constructor will be executed to Initialize that Object.
- Object Creation by New Operator and then Initialization by Constructor.



- Before Constructor Only Object is Ready and Hence within the Constructor we can Access Object Properties Like Hash Code.

```
class Test {
        Test() {
                System.out.println(this); // Test@6e3d60
                System.out.println(this.hashCode()); //7224672
        }
        public static void main(String[] args) {
                Test t = new Test();
        }
                                }
```

2. Whenever we are creating child class object whether Parent object will be created or not ????

- Whenever we are creating Child Class Object automatically Parent Constructor will be executed but Parent Object won't be created.
- The Purpose of Parent Constructor Execution is to Initialize Child Object Only. Of Course for the Instance Variables which are inheriting from parent Class.

```
class P {
        P() {
                System.out.println(this.hashCode()); //7224672
        }
}
class C extends P {
                C() {
                        System.out.println(this.hashCode()); //7224672
                }
}
class Test {
        public static void main(String[] args) {
                C c = new C();
                System.out.println(c.hashCode()); //7224672
        }
}
```

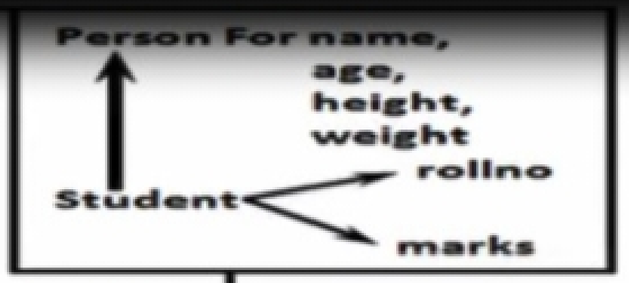
- In the Above Example whenever we are creating Child Object Both Parent and Child Constructors will be executed for Child Object Purpose Only.

3. Whenever we are creating child class object, what is the need of executing Parent class Constructor????

```
Person (String name, int age, int height, int weight)
{
        this.name = name;
        this.age = age;
        this.height = height;
        this.weight = weight;
}

Student (String name, int age, int height, int weight, int rollno, int marks)
{
        this.name = name;
        this.age = age;
        this.height = height;
        this.weight = weight;
        this.rollno = rollno;
        this.marks = marks;
}

Student s = new Student ("Ravi", 30, 6, 60, 101, 70);
```



name: Ravi
age: 30
height: 6
weight: 60
} These 4 Properties will be initialized by Parent Constructor

rollno: 101
marks: 70
} These 2 Properties will be initialized by Child Constructor

In the Above Example we are Just creating Student Object but Both Person and Student Constructor to Initialize Student Object Only

- Whenever we are creating an Object automatically Constructor will be executed but it May be One Constructor OR Multiple Constructors.

4. Anyway we cannot create object for abstract class either directly or indirectly, But abstract class can contain constructor. What is the need ????

- Either Directly OR Indirectly we can't Create Object for Abstract Class but Abstract Class can contain Constructor what is the Need.
- Whenever we are creating Child Object Automatically Abstract Class Constructor will be executed to Perform Initialization of Child Object for the Properties whether inheriting from Abstract Class (Code Reusability).

Without abstract Class Constructor

```
class Student extends Person {
        int rollno;
        int marks;
        Student(String name, int age, int height, int weight) {
                this.name = name;
                this.age = age;
                this.height = height;
                this.weight = weight;
                this.rollno = rollno;
                this.marks = marks;
        }
}
```

```
abstract class Person {
        String name;
        int age, height, weight;
}

class Teache extends Person {
        String subject;
        double salary;
}
```

5. Anyway we cannot create Object for Abstract class and interface. Abstract class can contain constructor but interface does not. Why ????

Any Way we can't Create Object for Abstract Class and Interface. But Abstract Class can contain Constructor but Interface doesn't Why?

- The Main Purpose of Constructor is to Perform Initialization of an Object i.e. to Provide Values for Instance Variables.
- Abstract Class can contain Instance Variables which are required for Child Class Object to Perform Initialization of these Instance Variables Constructor is required Inside Abstract Class.
- But Every Variable Present Inside Interface is Always public, static and final whether we are declaring OR Not and Every Interface Variable we should Perform Initialization at the Time of Declaration and Hence Inside Interface there is No Chance of existing Instance Variable.
- Due to this Initialization of Instance Variables Concept Not Applicable for Interfaces.
- Hence Constructor Concept Not required for Interface.

```
abstract class Person {
        String name;
        int age;
        Person(String name, int age) {
            this.name = name;
                this.age = age; → Current Child Class Object
        }
}
```

6. Inside Interface we can take Only Abstract Methods. But in Abstract Class Also we can take Only Abstract Methods Based on Our Requirement. Then what is the Need of Interface? i.e. Is it Possible to Replace Interface Concept with Abstract Class?

Inside Interface we can take Only Abstract Methods. But in Abstract Class Also we can take Only Abstract Methods Based on Our Requirement. Then what is the Need of Interface? i.e. Is it Possible to Replace Interface Concept with Abstract Class?

We can Replace Interface with Abstract Class but it is Not Good Programming Practice (This is Like requesting IAS Officer for sweeping Activity)

❶
```
abstract class X
{
    ----------
    ----------
}
class Test extends X {}
```

❷
```
interface X
{
    ----------
    ----------
}
class Test implements X {}
```

- While extending X we can't extend any Other Class.
- While implementing Interface we can extend any Other Class and Hence we won't Miss any Inheritance Benefit.

- In the Case ❶ Object Creation is Costly.
  Test t = new Test(); → 2 Mins

- In the Case ❷ Object Creation is Not Costly.
  Test t = new Test(); → 2 Sec

- Hence if everything is Abstract Highly Recommended to Use Interface but Not Abstract Class.
- If we are talking About Implementation of Course Partial Implementation then Only we should go for Abstract Class.