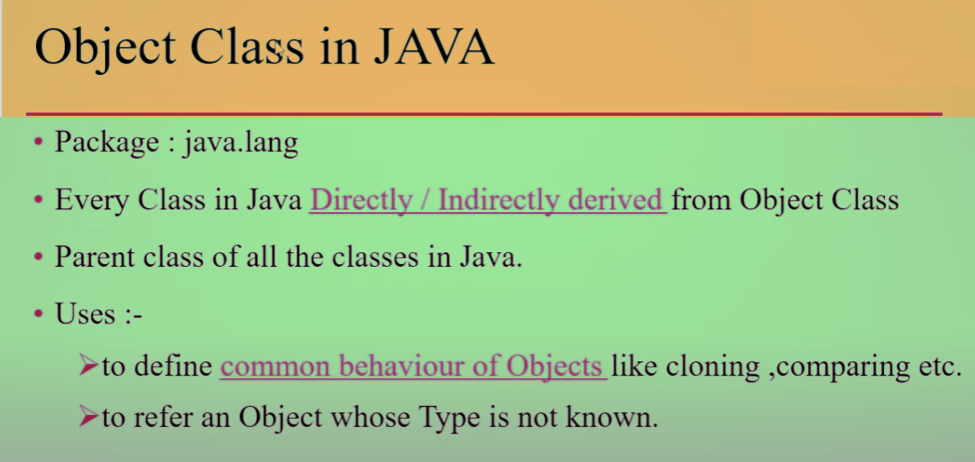
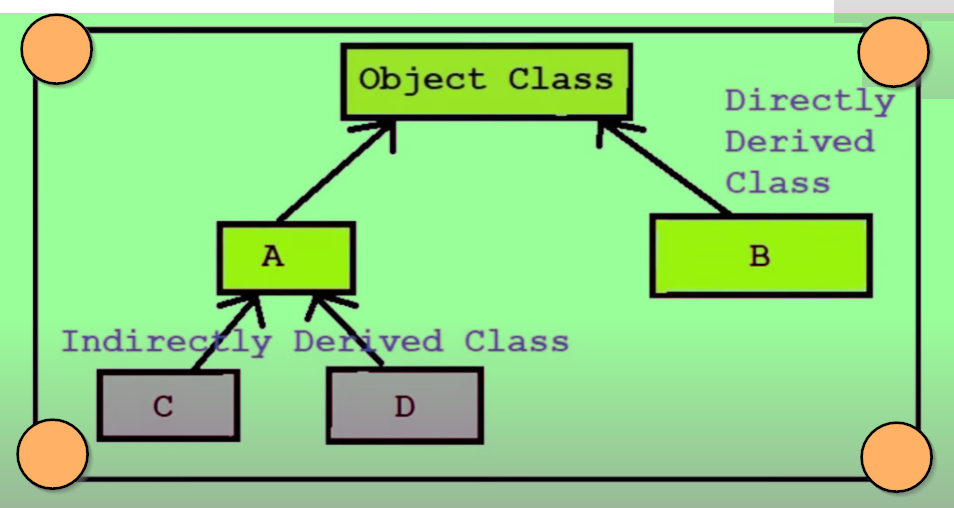
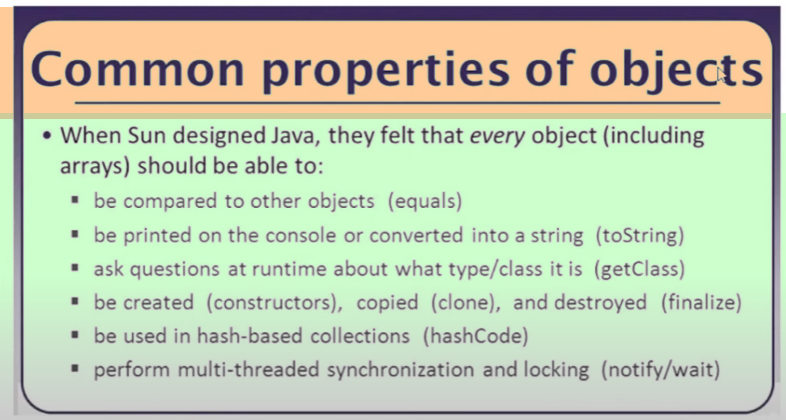
1. Object Class and Object Class Methods







🡪To refer an object whose Type is not known at runtime.

Let’s explain this with practical example :

/\* At runtime when we don't know about the return type of an object which we are passing to a method in that case we can use a object class method (i.e getClass() method).

And to compare our input object with a specific type (i.e Integer, Long, Float, String, etc) we can use the instanceof keyword in java.\*/

public class ObjectClassTest1 {

public static void checkObjectType(Object input) {

if(input **instanceof** Integer){

//instanceof is a keyword in Java that is used to test if an object is of a specified type.

System.out.println(input+" is an Integer type.");

}

else if(input **instanceof** Long){

System.out.println(input+" is a Long type.");

}

else if(input **instanceof** Float){

System.out.println(input+" is a Float type.");

}

else if(input **instanceof** String){

System.out.println(input+" is a String type.");

}

/\* else if(input **instanceof** Double){

System.out.println(input+" is a Double type.");

} \*/

else{

System.out.println(input+" is of "+input.getClass().getName()+" type....");

System.out.println(input+" is of "+input.getClass().getTypeName()+" type.");

}

}

public static void main(String[] args) {

checkObjectType(131); //int

checkObjectType(210L); //long

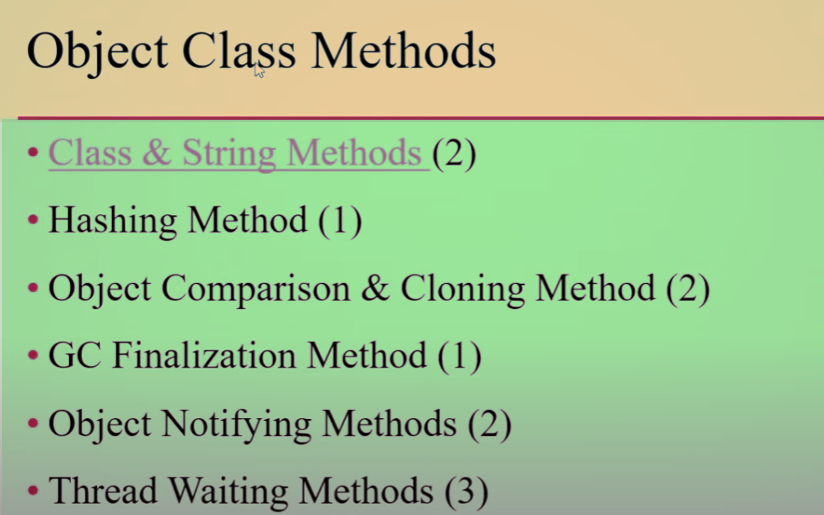
checkObjectType(11.5f); //float

checkObjectType("coding is my passion"); //String

checkObjectType(6.2d); //double

}

}



**Class & String Methods** **(2) :**

**public final class getClass()**

**public String toString()**

**Hashing Method (1) :**

**public int hashCode()**

**Object Comparison & Cloning Method (2) :**

**public boolean equals(Object obj)**

**protected object clone() throws CloneNotSupportedException**

**Garbage Collection finalization method (1) :**

**protected void finalize() throws Throwable**

**Objecy Notifying Methods (2) :**

**public final void notify()**

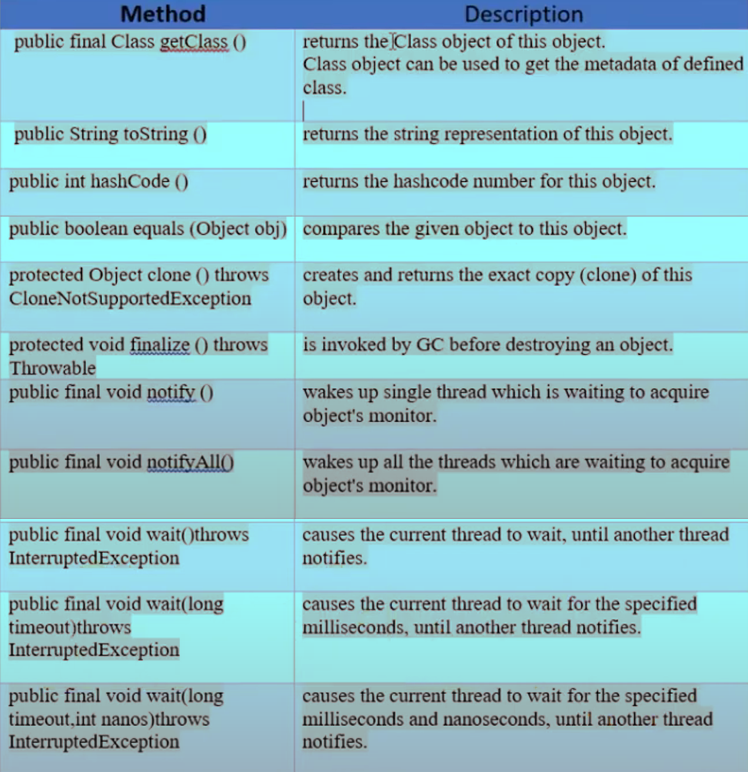
**public final void notifyAll()**

**Thread Waiting Methods (3) :**

**public final void wait() throws InterruptedException**

**public final void wait(long timeOUt) throws InterruptedException**

**public final void wait(long timeOut, int nanos) throws InterruptedException**



**Public final class getClass() :** Class object can be used to get the metadata of defined class.

Which means the name of the class and various properties and methods defined inside that class all these are metadata for the class.

It is used to obtain the runtime metadata of the object's class via the ***getClass().***

Being final, it cannot be overridden in any subclass.

**public String toString() :** If we want to represent any object as string then we use the toString().

**public int hashCode() :**

As we know hash code is a unique integer number that is used to uniquely represent an object in the memory.

So in java every object is having a unique hash code.

To get the unique value of the hash code we use hashcode() which is defined inside our object class which returns the integer value type.

**public Boolean equals (Object obj) :**

equals(-) and hashCode() are very important in hashing concept.

We can override equals(-) and hashCode() in our subclasses.

**protected Object clone() throws CloneNotSupportedException :**

When we are suppose to create an copy of an object then our object class must support cloning concept otherwise we get CloneNotSupportedException.

**protected void finalize() throws Throwable :**

Before destroying any object in the java programming language Garbage Collector calls the finalize() which present inside the object class to perform the cleanup activities.

***Here the below notify() and wait() are used to implement multithreading concepts***.

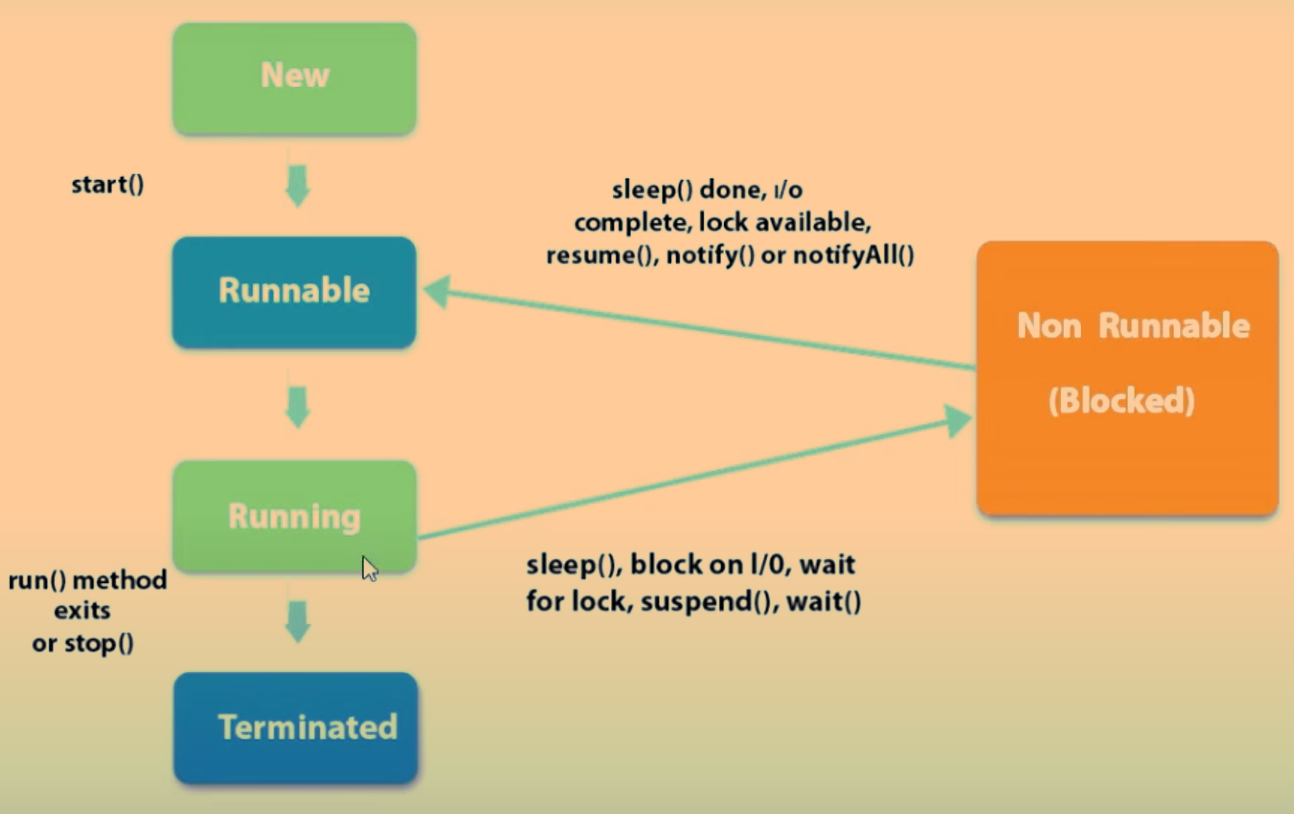
***public final void notify() :***

***public final void notifyAll() :***

***public final void wait() throws InterruptedException :***

***public final void wait(long timeOut) throws InterrruptedException :***

***public final void wait(long timeOut, in nanos) throws InterruptedException :***



Suppose let’s say one Thread which is currently in running state and when I call the **wait()** on the Running Thread in that case our Running Thread stop its execution and it will move into *Non Runnable State or* *Blocked State*.

And when we call a ***notify()/notifyAll()*** then in that case a Thread which is currently in Blocked/Non Runnable State will move into Runnable State.

So this ***notify()*** and ***wait*()** methods are used to implement MultiThreading concept in java programming language.

For suppose we want the current Thread to wait for a specified amount time then in that case we can call our ***wait(long timeout)*** parameter method.