Object Class

- Class are two types

  1 user defined

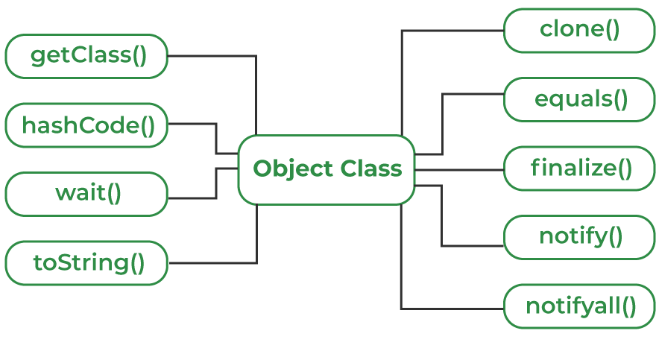
  2 pre-defined

- To create the object a class automatically to extend the object class

What is object class?

- Object class by default class in java. Object class is the parent class of all the classes.it is topmost class of java
- object class it is available from the java. lang. package
- object class is root class of the java programming
- entire the java technology either j2se and j2ee, j2me this is the top most root class is nothing but java. lang. object class

- objects class are contended are 11 methods
  1. public String to String ()
  2. public Boolean equals (Object o)
  3. public native int hash Code ()
  4. protected native Object clone () throws CloneNotSupportedException
  5. public final Class get Class ()
  6. protected void finalize () throws Throwable
  7. public final void wait () throws Interrupted Exception
  8. public final native void wait () throws Interrupted Exception
  9. public final void wait (long Ms, int ns) throws interrupted Exception
  10. public final native void notify ()
  11. public final native void notify All ()



why do we need object class?

- The Object class defines the basic state and behaviour that all objects must have, such as the ability to compare oneself to another object, to convert to a string, to wait on a condition variable, to notify other objects that a condition variable has changed, and to return the object's class.

How to use object class?

- The Object class provides multiple methods which are as follows:
- The to String() provides a String representation of an object and is used to convert an object to a String. ...
- It returns a hash value that is used to search objects in a collection.

where to use object class?

- The Object class is the parent class of all the classes in java by default. In other words, it is the topmost class of java.
- The Object class is beneficial if you want to refer any object whose type you don't know.
- Notice that parent class reference variable can refer the child class object, known as upcasting.

Main Points: -

- weight, notify and notify all method are thread related methods

how you are saying thread related methods?

- we are invoking that method comes in a exception is interpreted exception
- java is a thread based. If you are writing thread or not in your normal class runs in thread based only
- if the class is extending or not extends it can run thread based only
- every class in java running in the form of thread based

why do we need the finalize method?

- finalize method is calling before garbage the collection is calling
- why you are calling?
- to make deallocation with simple and make process is easy

All Method in Object Class: -

```
public class Book implements Cloneable {
   private String title;
   private String author;
   private int year;

   public Book(String title, String author, int year)
   {
      this.title = title;
      this.author = author;
      this.year = year;
   }
```

Override the toString method

```
   @Override public String toString()
   {
      return title + " by " + author + " (" + year + ")";
   }
```

Override the equals method

```
   @Override public boolean equals(Object obj)
   {
      if (obj == null || !(obj instanceof Book)) {
         return false;
      }
      Book other = (Book)obj;
```

```
  return this.title.equals(other.getTitle())
    && this.author.equals(other.getAuthor())
    && this.year == other.getYear();
}
```

Override the hash Code method

```
@Override public int hashCode()
{
  int result = 17;
  result = 31 * result + title.hashCode();
  result = 31 * result + author.hashCode();
  result = 31 * result + year;
  return result;
}
```

Override the clone method

```
@Override public Book clone()
{
  try {
    return (Book)super.clone();
  }
  catch (CloneNotSupportedException e) {
    throw new AssertionError();
  }
}
```

Override the finalize method

```
@Override protected void finalize() throws Throwable
{
   System.out.println("Finalizing " + this);
}

public String getTitle() { return title; }

public String getAuthor() { return author; }

public int getYear() { return year; }
public static void main(String[] args)
{
   // Create a Book object and print its details
   Book book1 = new Book(
      "The Hitchhiker's Guide to the Galaxy",
      "Douglas Adams", 1979);
   System.out.println(book1);

   // Create a clone of the Book object and print its
   // details
   Book book2 = book1.clone();
   System.out.println(book2);
```

Check if the two objects are equal

```
   System.out.println("book1 equals book2: "
                            + book1.equals(book2));

         Get the hash code of the two objects
        System.out.println("book1 hash code: "
                            + book1.hashCode());
        System.out.println("book2 hash code: "
                            + book2.hashCode());

        // Set book1 to null to trigger garbage collection
         and finalize method
        book1 = null;
        System.gc();
    }
}

class TestMain  {
    private int bullets = 40;

    // This method fires the number of bullets that are
    // passed it. When the bullet in magazine becomes zero,
    // it calls the wait() method and releases the lock.
    synchronized public void fire(int bulletsToBeFired)
    {
        for (int i = 1; i <= bulletsToBeFired; i++) {
            if (bullets == 0) {
                System.out.println(i - 1
                        + " bullets fired and "
                        + bullets + " remains");
            System.out.println(
                "Invoking the wait() method");
            try {
                wait();
            }
            catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println(
                "Continuing the fire after reloading");
        }

        bullets--;
    }
    System.out.println(
        "The firing process is complete");
}

// reload() increases the bullets by 40 everytime it is
// invoked and calls the notify() method which wakes up
// the thread that was sent to sleep using wait() inside
// of fire() method
synchronized public void reload()
{
        System.out.println(
            "Reloading the magazine and resuming "
            + "the thread using notify()");
        bullets += 40;
        notify();
    }
}

public class WaitDemo extends Thread {
    public static void main(String[] args)
    {

        TestMain  gf = new TestMain ();

        // Creating a new thread and invoking
        // our fire() method on it
        new Thread() {
            @Override public void run() { gf.fire(60); }
        }.start();
        // Creating a new thread and invoking
        // our reload method on it
        new Thread() {
            @Override public void run() { gf.reload(); }
        }.start();
    }
}
```