```
1 package com.pack1;
2
3 public class ClassB
4 {
5     public static void main(String[] args)
6     {
7         System.out.println(10+20);
8         System.out.println(["===>"+10+20]);
9     }
10 }
```

After the string data, whenever you are writing + symbol, it will work as concatenation operation.

```
30
 1 package com.pack1;
                                                           ===>1020
                                                           ===>30
 3 public class ClassB
 4 {
 58
       public static void main(String[] args)
 6
           System.out.println(10+20);
 7
           System.out.println("===>"+10+20);
8
           System.out.println("===>"+(10+20));
9
10
       }
11 }
12
```

### getClass()

### It is going to provide fully qualified class name

```
getClass() : class com.pack1.ClassA
1 package com.pack1;
 3 public class ClassA
 4 {
 5=
       public static void main(String[] args)
           ClassA aobj1=new ClassA();
8
           ClassA aobj2=new ClassA();
 9
10
           System.out.println("getClass() : "+aobj1.getClass());
11
12 }
13 /*
14 getClass(): It is going to provide fully qualified Class Nar
15
16 */
```

```
package com.pack1;

public class ClassA

{
    public static void main(String[] args)
    {
        ClassA aobj1=new ClassA();
        ClassA aobj2=new ClassA();
        System.out.println("aobj1 getClass() : "+aobj1.getClass());
        System.out.println("aobj2 getClass() : "+aobj2.getClass());
    }
}

// *

// getClass(): class com.pack1.ClassA aobj2 getClassA() : "the compact of the compact of the
```

\n is used to get a space line before in the console, this is a escape character. Our curser will be coming into the new line

### toString()

It is going to convert our class object into a String format

The internal implementation of toString()

La noming to The externition tries have drive to mindire is doi: to the lavoration to the conference of the conference o

```
247
         * value of:
248
         * <blockquote>
249
250
         * getClass().getName() + '@' + Integer.toHexString(hashCode())
251
         * </blockquote>
252
         * @return a string representation of the object.
253
254
255⊕
        public String toString() {
            return getClass().getName() + "@" + Integer.toHexString(hashCode());
256
257
258
259⊜
260
         * Wakes up a single thread that is waiting on this object's
261
         * monitor. If any threads are waiting on this object, one of them
         * is chosen to be awakened. The choice is arbitrary and occurs at
262
263
         * the discretion of the implementation. A thread waits on an object's
264
         * monitor by calling one of the {@code wait} methods.
265
266
         * The awakened thread will not be able to proceed until the current
267
         * thread relinquishes the lock on this object. The awakened thread will
         * compate in the usual manner with any other threads that might be
21
22
    toString(): It is going to convert Our Class Object into a String format
23
24
25
    public String toString()
26
27
            return getClass().getName()
                                                  Integer.toHexString(hashCode());
28
29
30 //
           com.pack1.ClassA@fdyd7gk
31
```

The number present at right last called as Hexa integer value of our class object

```
<terminated> ClassA [Java Application] C:\Program Files\Java\jdk-17\bin\javav.exe (20-F aobj1 getClass() : class com.pack1.ClassA
> 😅 Training > 🕮 src > 🏨 com.pack1 > 😭 ClassA > 💣 main(String[]) : void
              ClassA aobj1=new ClassA();
                                                                                               aobj2 getClass() : class com.pack1.ClassA
              ClassA aobj2=new ClassA();
                                                                                               aobj1 toString()) : com.pack1.ClassA@10f87f48
              System.out.println("aobj1 getClass() : "+aobj1.getClass());
                                                                                               aobj2 toString()) : com.pack1.ClassA@b4c966a
              System.out.println("aobj2 getClass(): "+aobj2.getClass());
              System.out.println("\naobj1 toString()) : "+aobj1.toString());
              System.out.println("aobj2 toString()) : "+aobj2.toString());
    getClass(): It is going to provide fully qualified Class Name
21
22
    toString(): It is going to convert Our Class Object into a String format
23
24
 25
     public String toString()
 26
 27
             return getClass().getName() + "@" + Integer.toHexString(hashCode(
 28
 29
     // aobj1 ===> com.pack1.ClassA@fdyd7gk
// aobi2 ===> com.nack.ClassA@Ausfu567
```

Because of .getName() we are not getting class

### finalize()

finalize() it will be called internally by the Garbage collector if there is an object eligible for destruction.

#### garbage collection

It is a process of destroying all unused or unreferenced objects from the heap area.

The garbage collector will be working in the background of every java program. You don't need to call it manually.

Whenever we are calling a garbage collector if there is an object eligible for destruction then only the garbage collector act on your program. Otherwise, it will just monitor the situation.

```
aobj1 getClass() : class com.pack1.ClassA
                                                                                                            aobj2 getClass() : class com.pack1.ClassA
 3 public class ClassA
 4 {
                                                                                                           aobj1 toString()) : com.pack1.ClassA@10f87f48
aobj2 toString()) : com.pack1.ClassA@b4c966a
         void meth1()
 58
               System.out.println("\nmeth1() called");
                                                                                                Line: 12 meth1() called
         public static void main(String[] args)
10
11
               ClassA aobj1=new ClassA();
12
              ClassA aobj2=new ClassA();
13
               System.out.println("aobj1 getClass() : "+aobj1.getClass());
System.out.println("aobj2 getClass() : "+aobj2.getClass());
14
15
16
               System.out.println("\naobj1 toString()) : "+aobj1.toString());
System.out.println("aobj2 toString()) : "+aobj2.toString());
17
18
19
20
               aobj1.meth1();
21
22
23
24
25 }
```

```
aobj1 getClass() : class com.pack1.ClassA
                                                                                         aobj2 getClass() : class com.pack1.ClassA
   public class ClassA
                                                                                         aobj1 toString()) : com.pack1.ClassA@10f87f48
       void meth1()
                                                                                         aobj2 toString()) : com.pack1.ClassA@b4c966a
            System.out.println("\nmeth1() called");
                                                                                         meth1() called
8
98
       public static void main(String[] args)
10
            ClassA aobj1=new ClassA();
12
            ClassA aobj2=new ClassA();
14
            System.out.println("aobj1 getClass() : "+aobj1.getClass());
            System.out.println("aobj2 getClass() : "+aobj2.getClass());
16
17
18
            System.out.println("\naobj1 toString()) : "+aobj1.toString());
System.out.println("aobj2 toString()) : "+aobj2.toString());
19
20
            aobj1.meth1();
            System.gc(); // It is a method to call Garbage Collector manually
25 }
```

You cannot perform any operations on null.

true, false and null are not key words, those are values.

```
3 public class ClassA
                                                                                                         aobj1 getClass() : class com.pack1.ClassA
                                                                                                         aobj2 getClass() : class com.pack1.ClassA
         void meth1()
                                                                                                         aobj1 toString()) : com.pack1.ClassA@10f87f48
              System.out.println("\nmeth1() called");
                                                                                                         aobj2 toString()) : com.pack1.ClassA@b4c966a
 8
         public static void main(String[] args)
                                                                                                         meth1() called
 10
              ClassA aobj1=new ClassA(); // 1st Object
12
              ClassA aobj2=new ClassA(); // 2nd Object
13
              System.out.println("aobj1 getClass() : "+aobj1.getClass());
System.out.println("aobj2 getClass() : "+aobj2.getClass());
14
15
16
17
              System.out.println("\naobj1 toString()) : "+aobj1.toString());
System.out.println("aobj2 toString()) : "+aobj2.toString());
18
19
 20
21
22
              aobj1.meth1();
              aobj1=null;
              System.gc(); // It is a method to call Garbage Collector manually
```

```
aobj1 getClass() : class com.pack1.ClassA
aobj2 getClass() : class com.pack1.ClassA
         void meth1()
              System.out.println("\nmeth1() called");
                                                                                                          aobj1 toString()) : com.pack1.ClassA@10f87f48
8
                                                                                                          aobj2 toString()) : com.pack1.ClassA@b4c966a
         protected void finalize()
10
                                                                                                          meth1() called
11
              System.out.println("Garbage has been collected");
12
13
         public static void main(String[] args)
14
15
              ClassA aobj1=new ClassA(); // 1st Object
16
             ClassA aobj2=new ClassA(); // 2nd Object
17
              System.out.println("aobj1 getClass()): "+aobj1.getClass());
System.out.println("aobj2 getClass() : "+aobj2.getClass());
18
19
20
              System.out.println("\naobj1 toString()) : "+aobj1.toString());
System.out.println("aobj2 toString()) : "+aobj2.toString());
21
22
23
24
              aobj1.meth1();
25
26
              //aobj1=null;
27
28
              System.gc(); // It is a method to call Garbage Collector manually
29
```

```
aobj1 getClass() : class com.pack1.ClassA
aobj2 getClass() : class com.pack1.ClassA
           void meth1()
6
7
8
9e
10
11
12
                 System.out.println("\nmeth1() called");
                                                                                                                              aobj1 toString()) : com.pack1.ClassA@10f87f48
aobj2 toString()) : com.pack1.ClassA@b4c966a
           protected void finalize()
                                                                                                                              meth1() called
Garbage has been collected
                 System.out.println("Garbage has been collected");
130
           public static void main(String[] args)
14
                 ClassA aobj1=new ClassA(); // 1st Object ClassA aobj2=new ClassA(); // 2nd Object
15
16
17
                 System.out.println("aobj1 getClass() : "+aobj1.getClass());
System.out.println("aobj2 getClass() : "+aobj2.getClass());
18
19
20
                 System.out.println("\naobj1 toString()) : "+aobj1.toString());
System.out.println("aobj2 toString()) : "+aobj2.toString());
21
22
23
                 aobj1.meth1();
24
25
                 aobj1=null;
26
27
                 System.gc(); // It is a method to call Garbage Collector manually
28
29
```

```
3 public class ClassA
 4 {
 5e
       void meth1()
 6
 7
           System.out.println("\nmeth1() called");
 8
9=
       protected void finalize()
10
11
           System.out.println("Garbage has been collected");
12
130
       public static void main(String[] args)
14
           ClassA aobj1=new ClassA(); // 1st Object
15
           ClassA aobj2=new ClassA(); // 2nd Object
16
                                                                       T
17
18
           System.out.println("aobj1 getClass() : "+aobj1.getClass());
           System.out.println("aobj2 getClass() : "+aobj2.getClass());
19
20
           System.out.println("\naobj1 toString()) : "+aobj1.toString());
21
22
           System.out.println("aobj2 toString()) : "+aobj2.toString());
23
24
           aobj1.meth1();
25
26
           aobj1=null;
           aobjl=null;
26
27
28
           System.gc(); // It is a method to call Garbage Collector manually
29
       }
30 }
31 /*
                 It is going to provide fully qualified Class Name
32 getClass():
33
34
35
    toString(): It is going to convert Our Class Object into a String format
36
37
38 public String toString()
39
           return getClass().getName() + "@" + Integer.toHexString(hashCode());
40
41
42
43 finalize():
44
    1) finalize() will be called internally by the Garbage Collector
45
46 if there is an Object eligible for destruction
47 */
```

# Object:

- Objects are created from a class
- To create an object of a Class, specify the class name, followed by the object/reference name, and use the keyword "new" (We can initialize that object by using constructors).
- We can create multiple objects of one class.

Note: **reference** is the address of the memory location where the object is stored

## Variable:

- A variable provides identity to memory location
- Using variables we can process the information easily
- Variables can also be called as References & Identifiers

#### **OBJECT**

- An object is created from a class.
- To create an object, first we need to write a new key word and initialize that object with the help of the constructor and constructor name should be same as class name.
- We can create multiple objects of a class.

#### **Variables**

- ❖ A variable provides identity to a memory location.
- Using variables, we can process information easily.

#### Naming Conventions of a Java Identifier:

1) A java identifier can start with

```
\rightarrow A to Z
```

a to z

\$ (or)

- 2) A Java identifier can **never** starts with a number, but we can use any number **combinations** between 0 to 9 in the identifiers.
- 3) Only '2' special characters are allowed in the identifiers \$ and
- 4) Spaces are NOT allowed in the identifiers.
- 5) We can take any length as identifier length.
- 6) As java is case-senstive int a=10; and int A=20; both are NOT SAME
- 7) All the Java language keywords we cant use them as Identifiers
- 8) We can use Class names as identifiers, but it is highly not recommended

public class ClassA

1

void meth1()

int x=10;

aobj.meth1();

System.out.println(x);

public static void main(String[] args)

ClassA aobj=new ClassA();

Identifier

### Understanding Identifier:

- A name in JAVA program is called identifier.
- It may be class name, method name, variable name.

#### Rules [8]:

- ✓ The only allowed characters in java identifiers are:
  - 1) a to z
  - 2) A to Z
  - 3) 0 to 9
  - 4) (underscore) and \$
- ✓ If we are using any other symbols we will get compile time error.
- ✓ We can't start a JAVA Identifier with number.
- ✓ Java Identifiers are case sensitive (Actually JAVA itself is a case sensitive)
- ✓ We can take our own length for the name of an JAVA identifier.
- ✓ We can't use JAVA language keywords (50) as identifiers.
- ✓ No space is allowed between the characters of an identifier
- ✓ All predefined JAVA class names and interface names we can use as identifiers (X)

```
int String =10;
String s="Java";
```

This an example for point 8

It is good programming practice that your identifiers' name should reflect their purpose.

All the 8 primitive data types are key words not valid for identifier name

