

Important keywords in java

note: keywords in java should always be in lower case:

1. class: Class is a factory that generates objects for us when ever it receives a request from "new

keyword"

Syntax:

class A {

}

2. new:

2.1 new keyword in java send a request to the class to create object

2.2 It gets the address of the object and stores that in reference variable

Syntax:

ClassName var = new ClassName();

static versus non static:

non static:

1. When ever an object is created non static member will be loaded into the object, where as static

members will never get loaded into the object

2. non static member can be accessed only after creating an Object

Example 1:

```
public class A {
```

```
    int i = 10;
```

```
    static int j = 100;
```

```
    public static void main(String args[]) {
```

```
        A a1 = new A();
```

```
        System.out.println(a1.i);
```

```
    }
```

```
}
```

Output:

10

Example 2:

```
public class A {
```

```
    int i = 10;
```

```
    int k = 100;
```

```
    static int j = 1000;
```

```
    public static void main(String args[]) {
```

```
        A a1 = new A();
```

```
        System.out.println(a1.i);
```

```
        System.out.println(a1.k);
```

```
    }
```

```
}
```

static:

1. Every class will have a dedicated common memory created on RAM. static members automatically gets

loaded into this common memory

2. To access static member we need not create any object

Example 1:

```
public class A {
```

```
    static int j = 1000;
```

```
    public static void main(String args[]) {
```

```
        System.out.println(A.j);
```

```
    }
```

```
}
```

Output:

1000

Example 2:

```
public class A {
```

```
    int i = 10; //non static
```

```
    static int j = 500; //static
```

```
    int k = 20; //non static
```

```
static int z = 1000;//static
```

```
public static void main(String args[]) {
```

```
    A a1 = new A();
```

```
    System.out.println(a1.i);
```

```
    System.out.println(a1.k);
```

```
    System.out.println(A.j);
```

```
    System.out.println(A.z);
```

```
}
```

```
}
```

Output

10

20

500

1000

Example 3:

```
public class A {
```

```
    static int i = 10;
```

```
    public static void main(String args[]) {
```

```
        A.i = 100;
```

```
        System.out.println(A.i);
```

```
    }
```

```
}
```

Output:

100

Static variables can be accessed in three ways:

1. ClassName.variableName
2. VariableName
3. ObjectReference.variableName

Example 1:

```
public class A {
```

```
static int i = 10;
```

```
public static void main(String args[]) {
```

```
System.out.println(A.i);
```

```
System.out.println(i); //A.i
```

```
A a1 = new A();
```

```
System.out.println(a1.i); //A.i
```

```
}
```

```
}
```

Output:

10

10

10

Overview Of Method():

What are non static methods:

These methods belong to Object, and only after creating the object we should be able to access this

method.

Example 1:

```
public class A {
```

```
    static int i = 10;
```

```
    public static void main(String args[]) {
```

```
        A a1 = new A();
```

```
        a1.test();
```

```
    }
```

```
    public void test() { // non static method
```

```
        System.out.println("From test");
```

```
    }
```

```
}
```


What are static methods:

These methods belongs to class and is accessed with the class name, here you need not create object

Example 2:

```
public class A {
```

```
    static int i = 10;
```

```
    public static void main(String args[]) {
```

```
        A.test();
```

```
    }
```

```
    public static void test(){//static method
```

```
        System.out.println("From test");
```

```
    }
```

```
}
```

Output:

From test

Types Of Variables in java:

1. static
2. non static
3. local variable
4. reference

1. Local variables

a. Local variables are created inside a method and should be used only within created methods. If you

use outside the method then it will throw error

b. We need not create object to access local variables

c. Local variables cannot be accessed without initializing it

d. var datatype was introduced in JDK 1.10 version, its a special datatype which can store any kind of value in it, but this variable has to local.

Example 1:

```
public class A {
```

```
    public static void main(String args[]) {
```

```
        A a1 = new A();
```

```
        a1.test();
```

```
        System.out.println(i);
```

```
}
```

```
public void test(){
```

```
int i = 10;
```

```
System.out.println(i);
```

```
}
```

```
}
```

Output: Error

Example 2:

```
public class A {
```

```
public static void main(String args[]) {
```

```
int i = 10;
```

```
System.out.println(i);
```

```
}
```

```
}
```

Output:

10

Example 3:

```
public class A {
```

```
    public static void main(String args[]) {
```

```
        int i ;
```

```
        System.out.println(i);
```

```
    }
```

```
}
```

Output:

Error

Example 4:

```
public class A {
```

```
    public static void main(String args[]) {
```

```
int i = 0 ; //main method
```

```
System.out.println(i); //main method
```

```
A.test();
```

```
}
```

```
public static void test(){
```

```
System.out.println(i); //Error
```

```
}
```

```
}
```

Output: Error

Example 5:

```
public class A {
```

```
public static void main(String args[]) {
```

```
var i = true;
```

```
var j = 100;
```

```
System.out.println(i);
```

```
System.out.println(j);
```

```
}
```

```
}
```

Output:

true

100

2. Static Variables:

a. static variables should be created always outside methods but inside a class using static keyword

b. It is non mandatory to initialize static variables, if we don't initialize then java compiler will

auto initialize with default value which depends on datatype

c. Static variables have global access

d. We need not create an object to access static variables

Example 1:

```
public class A {
```

```
public static void main(String args[]) {
```

```
    static int i = 100;
```

```
    //Error
```

```
    System.out.println(i);
```

```
}
```

```
}
```

Output: Error

Example 2:

```
public class A {
```

```
    static int i = 100;
```

```
    public static void main(String args[]) {
```

```
        System.out.println(A.i);
```

```
}
```

```
}
```

Output: 100

Example 3:

```
public class A {
```

```
    public static void main(String args[]) {
```

```
        System.out.println(A.i);
```

```
    }
```

```
    static int i = 100;
```

```
}
```

Output: 100

Example 4:

```
public class A {
```



```
static int i ;//0
```

```
public static void main(String args[]) {
```

```
System.out.println(A.i);
```

```
}
```

```
}
```

Output:

0

Example 5:

```
public class A {
```

```
static int i = 10;//Has gloabl Access
```

```
public static void main(String args[]) {
```

```
System.out.println(A.i);
```

```
A a1 = new A();
```

```
a1.test();
```

```
}
```

```
public void test(){//non static
```

```
System.out.println(A.i);
```

```
}
```

```
}
```

Output:

10

10

3. Non static variables:

- a. These variables belongs to object and can be accessed only after creating object
- b. These variables are created outside method but inside a class without static word
- c. It is non mandatory to initialize non static variables, if we donot initialize then java compiler

will auto initialize with with default value which is depends on datatype

Example 1:

```
public class A {
```

```
int i =100;
```

```
public static void main(String args[]) {
```

```
System.out.println(i);
```

```
// Error because it is non static variable and cannot be accessed without creating object
```

```
}
```

```
}
```

Output:

Error

Example 2:

```
public class A {
```

```
int i =100;
```

```
public static void main(String args[]) {
```

```
A a1 = new A();
```

```
System.out.println(a1.i);
```

```
}
```

```
}
```

Output:

100

Example 3:

```
public class A {
```

```
int i ;
```

```
public static void main(String args[]) {
```

```
A a1 = new A();
```

```
System.out.println(a1.i);
```

}

}

Output:

0

Installing JDK 1.8

Step 1:

Download JDK 1.8 version from internet (Most Popular and widely used version)

C>>Program file>>Java>> JDK / JRE

Step 2:

Down Eclipse for ee

ex: Kepler

oxygen

Shortcuts:

1. main --> press control + space bar + enter
2. Syso --> press Control + Space bar
3. control + 1-->

4. Reference variables: These are special variables in java that stores objects address

Syntax: `ClassName varName = new ClassName();`

```
public class A {  
  
    public static void main(String[] args) {  
        A a1 = new A();  
        System.out.println(a1);  
    }  
  
}
```

Output:

appja2.A@2a139a55

Types of reference variables:

1. local reference variables:

- a. these variables are created inside a method and should be used only within created method.
- b. local reference variables cannot be used without initializing it

Example 1:

```
public class A {  
  
    public static void main(String[] args) {  
        A a1 = new A();
```

```

        System.out.println(a1);

        System.out.println(a1);

        a1.test();

    }

    public void test() {

        System.out.println(a1);//Error

    }

}

```

Output: Error

Example 2:

```

public class A {

    public static void main(String[] args) {

        A a1 ;

        System.out.println(a1);//Error because a1 has no value stored in it

    }

}

```

Output: Error

2. static reference variables:

a. These variables are created outside all the methods but inside a class using static keyword. These

variables have global access

b. static reference variables if not initialized then automatically default value null will get stored

in it.

Example 1:

```
public class A {  
    static A a1 = new A();  
    public static void main(String[] args) {  
        System.out.println(a1);  
        a1.test();  
    }  
    public void test(){  
        System.out.println(a1);  
    }  
}
```

Output:

appja2.A@2a139a55

appja2.A@2a139a55

Example 2:

```
public class A {  
    static A a1; //null  
    public static void main(String[] args) {  
        System.out.println(a1);  
    }  
}
```


Output:

null

Type Casting: Converting a particular datatype into required datatype is called as type casting

Two type:

1. Auto Upcasting:

a. Converting a smaller datatype to bigger datatype is called as auto upcasting

b. During auto upcasting data loss should not happen.

Example 1:

```
package typecastingexamples;
```

```
public class A {
```

```
    public static void main(String[] args) {
```

```
        int i = 10; //4 bytes
```

```
        long j = i;
```

```
        System.out.println(j);
```

```
    }
```

```
}
```

Output:

10

Example 2:

```
public class A {
```

```
public static void main(String[] args) {  
  
    long i = 10; //8 bytes  
  
    int j = i;  
  
    System.out.println(j);  
  
}
```

Output: Error

Example 3:

```
public class A {
```

```
    public static void main(String[] args) {  
  
        byte i = 10; //8 bytes  
  
        short j = i;  
  
        System.out.println(j);  
  
    }  
  
}
```

Output:

10

Example 4:

```
package typecastingexamples;
```

```
public class A {
```

```
public static void main(String[] args) {  
    short i = 10; //8 bytes  
    byte j = i;  
    System.out.println(j);  
}  
  
}
```

Example 5:

```
public class A {
```

```
    public static void main(String[] args) {  
        float i = 10.3f; //8 bytes  
        double j = i;  
        System.out.println(j);  
    }  
  
}
```

Output:

10.3

Example 6:

```
int i ='a';
```

Output:97

Example 7:

```
public class A {  
  
    public static void main(String[] args) {  
  
        int i = 'ㄹ';  
  
        System.out.println(i);  
  
    }  
  
}
```

Output:

2319

Example 8:

```
public class A {  
  
    public static void main(String[] args) {  
  
        int i = 'ㄹ'+'ㄱ';  
  
        System.out.println(i);  
  
    }  
  
}
```

Output:

20010

Example 9:

```
public class A {  
  
    public static void main(String[] args) {  
  
        float i = 10.3f;  
  
        long j = i;  
  
        System.out.println(i);  
  
    }  
  
}
```

Output:

Error

Example 10:

```
package typecastingexamples;
```

```
public class A {  
  
    public static void main(String[] args) {  
  
        int i = 'a'+ 'b' + 'c';//97//98/99  
  
        System.out.println(i);  
  
    }  
  
}
```

Output:

294

2. Explicit Downcasting

- a. Converting bigger datatype to smaller datatype is called as downcasting
- b. When during conv. if data loss happens then it is called downcasting

Example 1:

```
package typecastingexamples;
```

```
public class A {
```

```
    public static void main(String[] args) {
```

```
        long i = 10;
```

```
        int j = (int) i;
```

```
        System.out.println(j);
```

```
    }
```

```
}
```

Output:

10

Example 2:

```
public class A {
```

```
    public static void main(String[] args) {
```

```
        int i = 10;
```

```
        byte j = (byte) i;
```

```
        System.out.println(j);  
    }  
}
```

Output:

10

Example 3:

```
public class A {
```

```
    public static void main(String[] args) {  
        double i = 10.3;  
        float j = (float) i;  
        System.out.println(j);  
    }  
}
```

Ouput:

10.3

Example 4:

```
public class A {
```

```
    public static void main(String[] args) {  
        float i = 10.3F;  
    }  
}
```

```
        long j = (long)i;  
        System.out.println(j);  
    }
```

```
}
```

Output:

10

Pankaj Sir Academy 9632882052