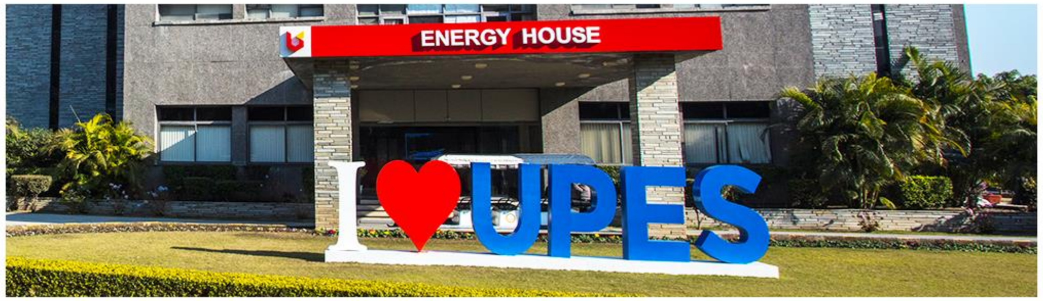

UPES

UNIVERSITY WITH A PURPOSE

Object Oriented Programming

Can we Overload or Override static methods in java ?

Overriding:

- Overriding is a feature of OOP languages like Java that is related to run-time polymorphism. A subclass (or derived class) provides a specific implementation of a method in the superclass (or base class).
- The implementation to be executed is decided at run-time and a decision is made according to the object used for the call. Note that signatures of both methods must be the same.

Overloading:

- Overloading is also a feature of OOP languages like Java that is related to compile-time (or static) polymorphism.
- This feature allows different methods to have the same name, but different signatures, especially the number of input parameters and type of input parameters.

Can we overload static methods?

‘Yes’. We can have two or more static methods with the same name, but differences in input parameters. For example, consider the following Java program.

```
// filename Test.java
public class Test {
        public static void foo() {
                System.out.println("Test.foo() called ");
        }
        public static void foo(int a) {
                System.out.println("Test.foo(int) called ");
        }
        public static void main(String args[])
        {
                Test.foo();
                Test.foo(10);
        }
}
```

Output

```
Test.foo() called
Test.foo(int) called
```

Can we overload methods that differ only by static keyword?

We cannot overload two methods in Java if they differ only by static keyword (number of parameters and types of parameters is the same).

```
// filename Test.java
public class Test {
        public static void foo() {
                System.out.println("Test.foo() called ");
        }
        public void foo() { // Compiler Error: cannot redefine foo()
                System.out.println("Test.foo(int) called ");
        }
        public static void main(String args[]) {
                Test.foo();
        }
}
```

Compiler Error, cannot redefine foo()

Can we Override static methods in java?

- We can declare static methods with the same signature in the subclass, but it is not considered overriding as there won't be any run-time polymorphism. Hence the answer is **'No'.**
- If a derived class defines a static method with the same signature as a static method in the base class its definition gets hidden and remains same as of parent class

Can we Override static methods in java?

```
/* Java program to show that if static method is redefined by
a derived class, then it is not overriding. */

// Superclass
class Base {

// Static method in base class which will be hidden in subclass
        public static void display() {
        System.out.println("Static or class method from
Base");    }
// Non-static method which will be overridden in derived
class
        public void print() {
        System.out.println("Non-static or Instance method
from Base");
        }   }
// Subclass
class Derived extends Base {
// This method hides display() in Base
        public static void display() {
        System.out.println("Static or class method from
Derived");          }
// This method overrides print() in Base
public void print() {
        System.out.println("Non-static or Instance
method from Derived");
}
}
// Driver class
public class Test {
public static void main(String args[ ]) {
Base obj1 = new Derived();

// As per overriding rules this should call to class
Derived's static
// overridden method. Since static method can not be
overridden, it
// calls Base's display()
obj1.display();

// Here overriding works and Derive's print() is called
obj1.print();
        }  }
```

UPES
UNIVERSITY WITH A PURPOSE

Can we Override static methods in java?

The following are some important points for method overriding and static methods in Java.

1. For class (or static) methods, the method according to the type of reference is called, not according to the object being referred, which means method call is decided at compile time.
2. For instance (or non-static) methods, the method is called according to the type of object being referred, not according to the type of reference, which means method calls is decided at run time.
3. An instance method cannot override a static method, and a static method cannot hide an instance method.
4. In a subclass (or Derived Class), we can overload the methods inherited from the superclass. Such overloaded methods neither hide nor override the superclass methods — they are new methods, unique to the subclass.

References

http://docs.oracle.com/javase/tutorial/java/IandI/override.html