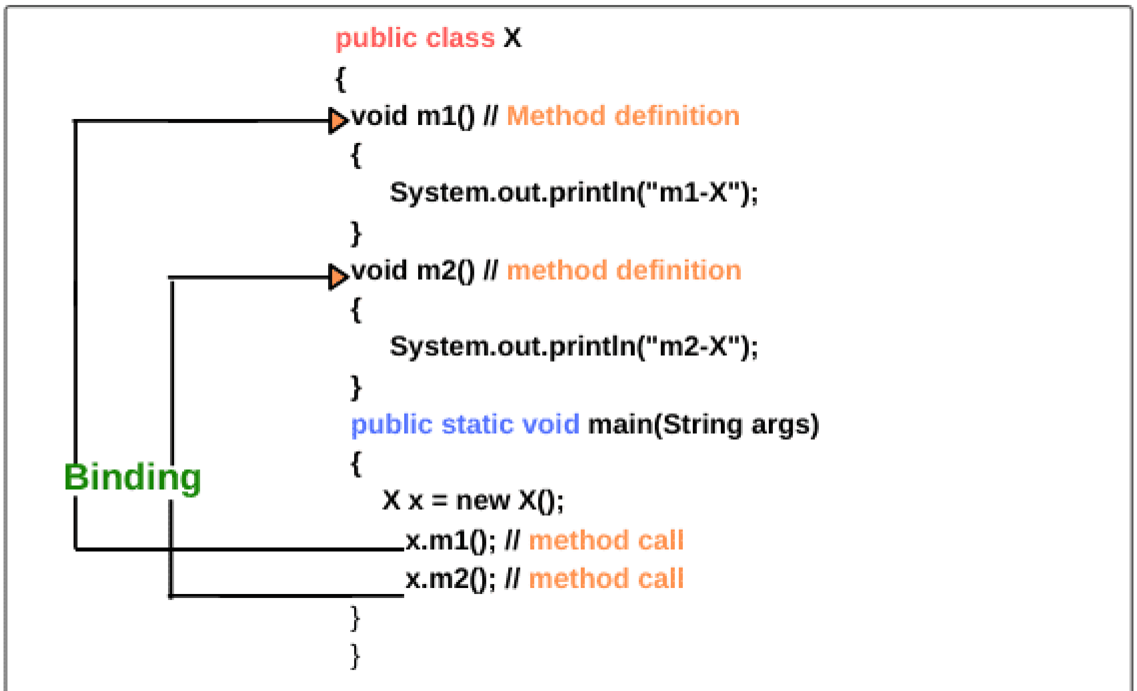
Static Binding and Dynamic Binding

The connecting (linking) between a method call and method body/definition is called **binding in java**.



```
public class X
{
 void m1() // Method definition
 {
     System.out.println("m1-X");
 }
 void m2() // method definition
 {
     System.out.println("m2-X");
 }
 public static void main(String args)
 {
     X x = new X();
       x.m1(); // method call
       x.m2(); // method call
 }
 }
```

Fig: Binding between method call and method definition

Types of Binding in Java

There are two types of binding in java. They are as follows:

1. **Static Binding** (also known as Early Binding).

2. **Dynamic Binding** (also known as Late Binding).

Static Binding in Java

The binding that happens during compilation is called **static binding in java**. This binding is resolved at the compiled time by the compiler.

In static binding, the java compiler does not check the type of object to which a particular reference variable is pointing to it.

Java compiler just checks which method is going to be called through reference variable and method definition exists or not.

This binding is also known as **early binding** because it takes place before the program actually runs. An example of static binding is *method overloading*.

```
package methodOverloading;
public class Addition
{
void add(int x, int y)
{
 int sum = x + y;
System.out.println("Sum of two numbers: " +sum);
 }
void add(int x, int y, int z)
{
 int sum = x + y + z;
 System.out.println("Sum of three numbers: " +sum);
 }
public static void main(String[] args)
{
  Addition a = new Addition();
  a.add(10, 20);
// Calling add() method with passing two argument values.
  a.add(20, 30, 40);
 // Calling add() method with passing three argument values.
 }
}
```

In the above preceding code, Java compiler did not check the type of object. It just checked only method call through reference variable and method definition.

If there is any private, final, or static method in a class, all these method binding are called static binding. For example:

```
public class Lion
{
private void eat()
{
 System.out.println("Lion eats flesh");
 }
static void live()
{
 System.out.println("Lion lives in Jungle");
}
public static void main(String[] args)
{
 Lion l = new Lion();
 l.eat();
 Lion.live();
 }
}
```

Dynamic Binding in Java

The binding which occurs during runtime is called dynamic binding in java. This binding is resolved
based on the type of object at runtime. In dynamic binding, the actual object is used for binding at runtime.

Dynamic binding is also called late binding or runtime binding because binding occurs during
runtime. An example of dynamic binding is *method overriding*.

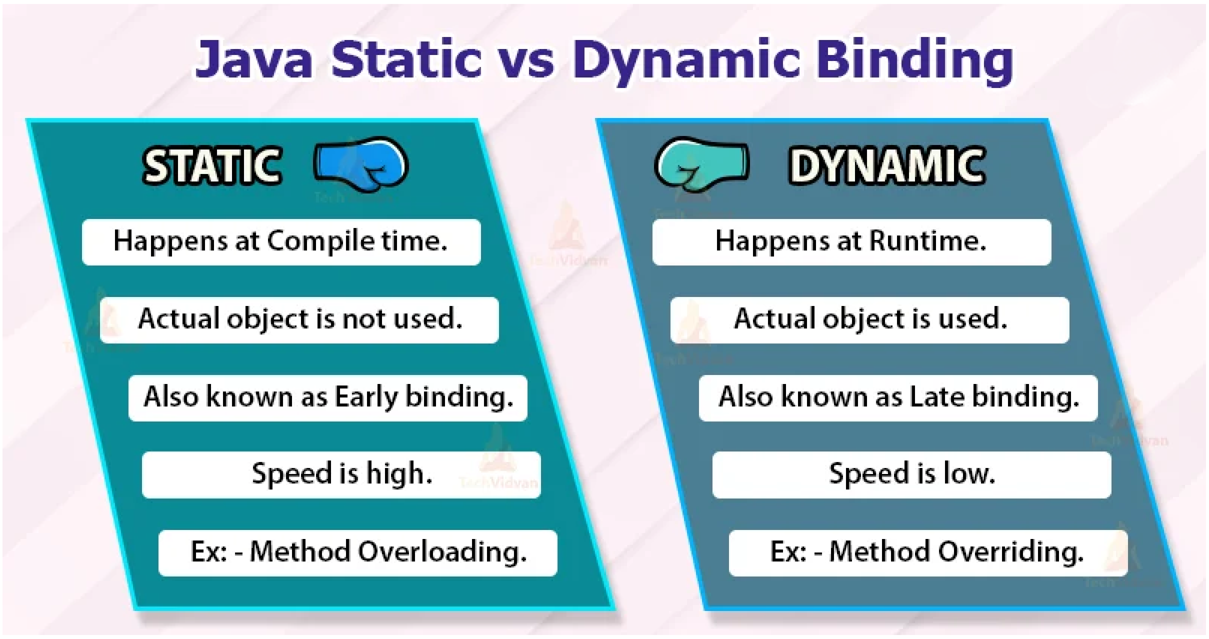
```
package dynamicBinding;
public class Animal
{
void eat()
{
 System.out.println("Animals eat both plants and flesh");
 }
}
public class Lion extends Animal
{
void eat()
{
 System.out.println("Lions eat flesh because they are carnivore");
 }
}
public class DynamicBindingTest
{
 public static void main(String[] args)
 {
  Animal a = new Animal();
   a.eat(); // It will call eat() method of Animal class
// because the reference variable is pointing to object of animal class.
  Animal a1 = new Lion();
   a1.eat(); // It will call eat() method of Lion class
//because the reference variable is pointing towards the object of Loin class.
 }
}
```

In the preceding example, object type cannot be determined by the compiler. Therefore, JVM resolved the method calls based on the type of object at runtime.

Why binding of private, static, and final methods are always static binding?

Static binding is better performance-wise because java compiler knows that all such methods cannot be overridden and will always be accessed by object reference variable.

Hence, the compiler doesn't have any difficulty in binding between a method call and method definition. That's why binding for such methods is always static.


Java Static vs Dynamic Binding
STATIC
Happens at Compile time.
Actual object is not used.
Also known as Early binding.
Speed is high.
Ex: - Method Overloading.
DYNAMIC
Happens at Runtime.
Actual object is used.
Also known as Late binding.
Speed is low.
Ex: - Method Overriding.

Difference between Static and Dynamic Binding in Java

There are the following differences between static and dynamic binding. They are as follows:

1. Static binding occurs at compile-time while dynamic binding occurs at runtime.

2. In static binding, actual object is not used whereas, actual object is used in the dynamic binding.

3. Static binding is resolved at compile time whereas, dynamic binding is resolved at runtime.

4. Static binding is also called early binding because it happens during compilation whereas, dynamic binding is called late binding because it happens during runtime.

5. An example of static binding is method overloading whereas, the example of dynamic binding is method overriding.

6.Private, static, and final methods show static binding because they cannot be overridden whereas, except private, static, and final methods,other methods show dynamic binding because they can be overridden.