Static and dynamic binding in java

BY CHAITANYA SINGH | FILED UNDER: OOPS CONCEPT

Association of method call to the method body is known as binding. There are two types of binding: **Static Binding** that happens at compile time and **Dynamic Binding** that happens at runtime. Before I explain static and dynamic binding in java, lets see few terms that will help you understand this concept better.

What is reference and object?

```
class Human{
....
}
class Boy extends Human{
   public static void main( String args[]) {
        /*This statement simply creates an object of class
         *Boy and assigns a reference of Boy to it*/
        Boy obj1 = new Boy();

        /* Since Boy extends Human class. The object creation
         * can be done in this way. Parent class reference
         * can have child class reference assigned to it
         */
        Human obj2 = new Boy();
   }
}
```

Static and Dynamic Binding in Java

As mentioned above, association of method definition to the method call is known as binding. There are two types of binding: Static binding and dynamic binding. Lets discuss them.

Static Binding or Early Binding

The binding which can be resolved at compile time by compiler is known as static or early binding. The binding of static, private and final methods is compile-time. **Why?** The reason is that the these method cannot be overridden and the type of the class is determined at the compile time. Lets see an example to understand this:

Static binding example

Here we have two classes Human and Boy. Both the classes have same method walk() but the method is static, which means it cannot be overriden so even though I have used the object of Boy class while creating object obj, the

parent class method is called by it. Because the reference is of Human type (parent class). So whenever a binding of static, private and final methods happen, type of the class is determined by the compiler at compile time and the binding happens then and there.

```
class Human{
   public static void walk()
   {
       System.out.println("Human walks");
   }
}
class Boy extends Human{
   public static void walk(){
       System.out.println("Boy walks");
   }
   public static void main( String args[]) {
       /* Reference is of Human type and object is
        * Boy type
        */
       Human obj = new Boy();
       /* Reference is of HUman type and object is
        * of Human type.
        */
       Human obj2 = new Human();
       obj.walk();
       obj2.walk();
   }
}
```

Output:

```
Human walks
Human walks
```

Dynamic Binding or Late Binding

When compiler is not able to resolve the call/binding at compile time, such binding is known as Dynamic or late Binding. Method Overriding is a perfect example of dynamic binding as in overriding both parent and child classes have same method and in this case the **type of the object**determines which method is to be executed. The type of object is determined at the run time so this is known as dynamic binding.

Dynamic binding example

This is the same example that we have seen above. The only difference here is that in this example, overriding is actually happening since these methods are **not** static, private and final. In case of overriding the call to the overriden method is determined at runtime by the type of object thus late binding happens. Lets see an example to understand this:

```
class Human{
   //Overridden Method
```

```
    public void walk()
    {
        System.out.println("Human walks");
    }
}
class Demo extends Human{
    //Overriding Method
    public void walk(){
        System.out.println("Boy walks");
    }
    public static void main( String args[]) {
        /* Reference is of Human type and object is
         * Boy type
         */
        Human obj = new Demo();
        /* Reference is of HUman type and object is
         * of Human type.
         */
        Human obj2 = new Human();
        obj.walk();
        obj2.walk();
    }
}
```

Output:

```
Boy walks
Human walks
```

As you can see that the output is different than what we saw in the static binding example, because in this case while creation of object obj the type of the object is determined as a Boy type so method of Boy class is called. Remember the type of the object is determined at the runtime.

Static Binding vs Dynamic Binding

Lets discuss the **difference between static and dynamic binding in Java**.

1. Static binding happens at compile-time while dynamic binding happens at runtime.
2. Binding of private, static and final methods always happen at compile time since these methods cannot be overridden. When the method overriding is actually happening and the reference of parent type is assigned to the object of child class type then such binding is resolved during runtime.
3. The binding of overloaded methods is static and the binding of overridden methods is dynamic.