**Static and dynamic binding in java**

[**OOPS CONCEPT**](http://beginnersbook.com/category/oops-concept/)

Before I explain **static and dynamic binding in java**, lets discuss few terms which will help you to understand the concepts 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 point to a child class object\*/

Human obj2 = new Boy();

}

}

**Static and Dynamic Binding in Java**

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 one by one.

**Static Binding or Early Binding**

The binding which can be resolved at compile time by compiler is known as static or early binding. All the static, private and final methods have always been bonded at [**compile-time**](http://beginnersbook.com/2013/04/runtime-compile-time-polymorphism/) . **Why binding of Static, final and private methods is always a static binding?**You would understand it better after reading dynamic binding. Still let me explain this – Compiler knows that all such methods cannot be overridden and will always be accessed by object of local class. Hence compiler doesn’t have any difficulty to determine object of class (local class for sure). That’s the reason binding for such methods is static.

**Static binding example**

class Human{

....

}

class Boy extends Human{

public void walk(){

System.out.println("Boy walks");

}

public static void main( String args[]) {

Boy obj1 = new Boy();

obj1.walk();

}

}

Here we have created an object of Boy class and calling the method walk()of the same class. Since nothing is ambiguous here, compiler would be able to resolve this binding during compile-time, such kind of binding is known as static binding.

**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. Overriding is a perfect example of dynamic binding as in overriding both parent and child classes have same method. Thus while calling the overridden method, the compiler gets confused between parent and child class method(since both the methods have same name).

**Dynamic binding example**

Here Human is a super class and Boy is a child class since Boy extends Human. Both these classes have a same method void walk(). Since we have assigned the parent class reference to the child class object, during call ofwalk() method the compiler would not be able to decide which walk()method to call. It will be confused between the walk() method of Human class and walk() method of Boy class. Such kind of bindings are known as dynamic binding as compiler figures out the object type in runtime.

package beginnersbook.com;

class Human{

public void walk()

{

System.out.println("Human walks");

}

}

class Boy extends Human{

public void walk(){

System.out.println("Boy walks");

}

public static void main( String args[]) {

//Reference is of parent class

Human myobj = new Boy();

myobj.walk();

}

}

Output:

Boy walks

**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. Binding of overridden methods happen at runtime.
3. Java uses static binding for [**overloaded methods**](http://beginnersbook.com/2013/05/method-overloading/) and dynamic binding for overridden methods.