In Java, member variables have static binding because Java does not allow for polymorphic behavior with member variables.

Here are few important difference between static and dynamic binding

1. Static binding in Java occurs during compile time while dynamic binding occurs during runtime.
2. private, final and static methods and variables use static binding and are bonded by compiler while virtual methods are bonded during runtime based upon runtime object.
3. Static binding uses Type (a class in Java) information for binding while dynamic binding uses object to resolve binding.
4. Overloaded methods are bonded using static binding while overridden methods are bonded using dynamic binding at runtime.

Here is an example which will help you to understand both static and dynamic binding in Java.

**Static Binding Example in Java**

public class StaticBindingTest {

public static void main(String args[]) {

Collection c = new HashSet();

StaticBindingTest et = new StaticBindingTest();

et.sort(c);

}

//overloaded method takes Collection argument

public Collection sort(Collection c) {

System.out.println("Inside Collection sort method");

return c;

}

//another overloaded method which takes HashSet argument which is sub class

public Collection sort(HashSet hs) {

System.out.println("Inside HashSet sort method");

return hs;

}

}

**Output**: Inside Collection sort method

**Example of Dynamic Binding in Java**

public class DynamicBindingTest {

public static void main(String args[]) {

Vehicle vehicle = new Car(); //here Type is vehicle but object will be Car

vehicle.start(); //Car's start called because start() is overridden method

}

}

class Vehicle {

public void start() {

System.out.println("Inside start method of Vehicle");

}

}

class Car extends Vehicle {

@Override

public void start() {

System.out.println("Inside start method of Car");

}

}

**Output:** Inside start method of Car