Core Java – 2021 – 2022

**Static Binding VS Dynamic Binding**

**Binding** refers to the link between method call and method definition.

1. Static Binding (also known as Early Binding). When **type of object is determined at compile time**
2. Dynamic Binding (also known as Late Binding). When **type of object is determined at run time**

Static binding uses Type(Class in Java) information for binding while Dynamic binding uses Object to resolve binding. If there is any private, final or static method in a class, there is static binding.

**Example of static binding**

**class** Dog {

**private** **void** eat() {Sop("dog is eating...");}

**public** **static** **void** main(String args[]) {

Dog d1=**new** Dog();

d1.eat();

 }

}

**Example of dynamic binding**

**class** Animal {

**void** eat() {Sop("animal is eating...");}

}

**class** Dog **extends** Animal {

**void** eat() {System.out.println("dog is eating..."); }

**public** **static** **void** main(String args[]) {

Animal a=**new** Dog();

a.eat();  🡺 Output:dog is eating...

}

}

|  |  |
| --- | --- |
| **Static Binding** | **Dynamic Binding** |
| It is a binding that happens at compile time. | It is a binding that happens at run time. |
| Actual object is not used for binding. | Actual object is used for binding. |
| It is also called early binding because binding happens during compilation. | It is also called late binding because binding happens at run time. |
| Method overloading is the best example of static binding. | Method overriding is the best example of dynamic binding. |
| Private, static and final methods show static binding. Because, they can not be overridden. | Other than private, static and final methods show dynamic binding. Because, they can be overridden. |

**AutoCloseable – Try with Resouce**

We have a new super interface **[java.lang.AutoCloseable](http://docs.oracle.com/javase/7/docs/api/java/lang/AutoCloseable.html" \o "AutoCloseable)**. This interface have one method:

**void close() throws Exception;**

**Code prior to Autoclosable interface**

BufferedReader br = **null**;  
 **try** {  
 String sCurrentLine;  
 br = **new** BufferedReader(**new** FileReader(**"C:/temp/test.txt"**));  
 **while** ((sCurrentLine = br.readLine()) != **null**) {  
 System.***out***.println(sCurrentLine);  
 }  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 } **finally** {  
 **try** {  
 **if** (br != **null**) **br.close()**;  
 } **catch** (IOException ex) {  
 ex.printStackTrace();  
 }  
 }

**With Autoclosabe, new way of writing**

**try (BufferedReader br = new BufferedReader(new FileReader("C:/temp/test.txt")))** {  
 String sCurrentLine;  
 while ((sCurrentLine = br.readLine()) != null) {  
 System.out.println(sCurrentLine);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }

Java docs recommend this interface to be **implemented on any resource that must be closed when it is no longer needed**. When we open any such AutoCloseable resource in special try-with-resource block, immediately after finishing the try block, **JVM calls this close() method on all resources initialized in “try()” block**.

You can also implement the above interface for cleaning operations.

**public class** MyClosbale **implements** AutoCloseable {  
  
 **public** String getSomeInfo() {  
 **return "some info"**;  
 }  
  
 @Override  
 **public void** close() **throws** Exception {  
 System.***out***.println(**"Getting closed ..."**);  
 }  
}

**public class** Test {  
 **public static void** main(String[] args) **throws** Exception {  
 **try**(MyClosbale cl = **new** MyClosbale()) {  
 System.***out***.println(cl.getSomeInfo());  
 }  
 }  
}

**OUTPUT**

some info

Getting closed ...

**public static void** main(String[] args) **throws** Exception {  
 MyClosbale cl = **new** MyClosbale();  
 System.***out***.println(cl.getSomeInfo());  
}

**OUTPUT**

some info

**\*\* In the above close method is never called, it means, it gets called when you provide try() block.**

But if you simply write like this,

You can also write like this.

**try (CustomResource cr = new CustomResource()) {**

**cr.accessResource();**

**}**

When we open any such AutoCloseable resource in special try-with-resource block, immediately after finishing the try block, **JVM calls this close() method on all resources initialized in “try()” block.**