**9. Compare and contrast**

**a. static and dynamic binding and identify usage of each**

1. Static Binding

The binding in Java language that can be resolved at the compile time by the compiler itself is known as the static or early binding. Generally, binding of all the private, static, and the final methods are done at the compile-time. In general, the static binding delivers a better performance as there is no extra overhead required.

|  |
| --- |
| example:  public class BaseClass {  public static class superclass  {  static void print()  {  System.out.println("print in superclass.");  }  }  public static class subclass extends superclass  {  static void print()  {  System.out.println("print in subclass.");  }  }  public static void main(String[] args)  {  superclass sup = new superclass();  superclass sub = new subclass();  sup.print();  sub.print();  }  }  Output:  print in superclass.  print in subclass. |

2. Dynamic Binding in Java

In the dynamic binding in Java language the compiler isn’t the one that decide the method to be called. The finest example of the dynamic binding is the overriding. What generally happens in the overriding is that both the parent and child classes have the exact same method. In very simple words, it can be understood as; whenever the type of the object is generally determined at the run-time then it is called as the dynamic binding in the Java.

|  |
| --- |
| Example:  public class BaseClass {  public static class superclass  {  void print()  {  System.out.println("print in superclass.");  }  }  public static class subclass extends superclass  {  @Override  void print()  {  System.out.println("print in subclass.");  }  }  public static void main(String[] args) {  superclass sup = new superclass();  superclass sub = new subclass();  sup.print();  sub.print();  } }  Output: print in superclass.  print in subclass. |