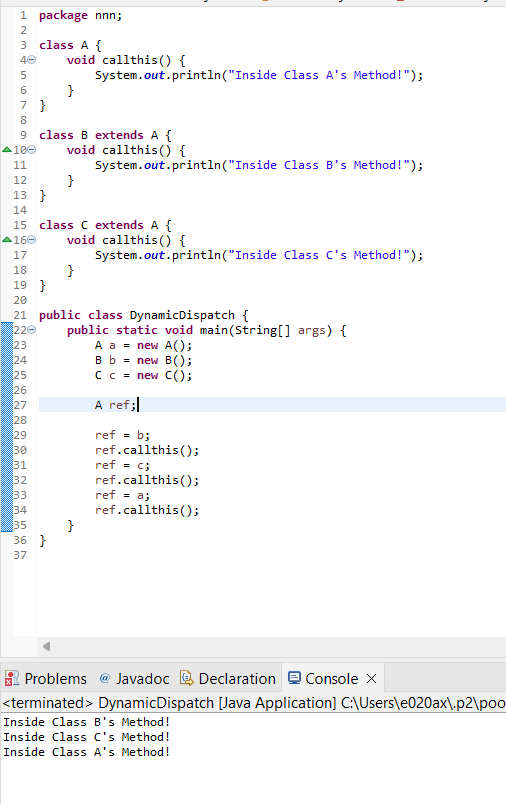
**SECTION \_7.5**

****

**2.Key Differences**

* **Abstract Class:**
  + Can provide a partial implementation.
  + Can hold state (instance variables).
  + Supports constructor.
  + A class can only extend one abstract class.
* **Interface:**
  + Cannot provide an implementation (except for default and static methods).
  + Cannot hold state (fields are constants).
  + Does not support constructors.
  + A class can implement multiple interfaces.

**Choosing Between Abstract Class and Interface**

* Use an abstract class when you need to share code among closely related classes and also need to enforce some methods to be implemented by subclasses.
* Use an interface when you want to define a contract that can be implemented by classes from different class hierarchies, or when you need to support multiple inheritance of type.

3.

public interface A {

void methodA();

}

public class B implements A {

@Override

public void methodA() {

// Implementation of methodA

}

}

public abstract class C {

abstract void methodC();

}

public class D extends C {

@Override

void methodC() {

// Implementation of methodC

}

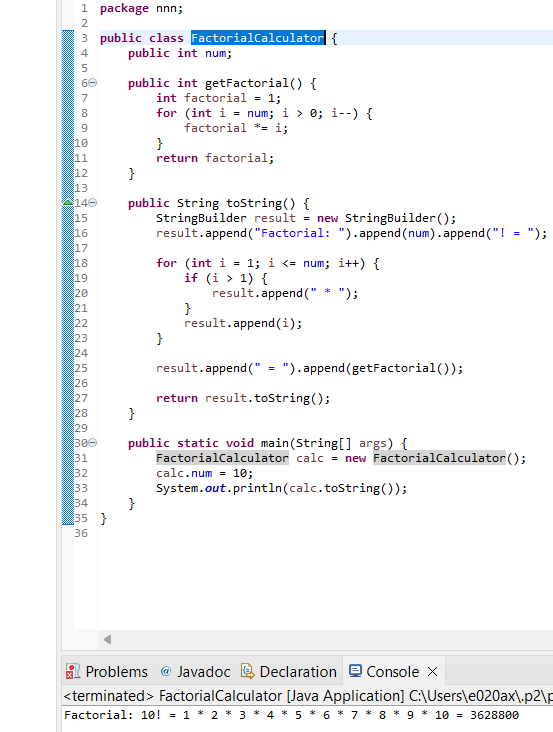
}

public class E extends B {

// No additional methods or fields

|  |  |
| --- | --- |
| code | Always Compile, Sometimes Compile, or Does Not Compile? |
| a = new B(); | Always Compile |
| d = new C(); | Does Not Compile |
| b.methodA(); | Always Compile |
| e.methodA(); | Always Compile |
| c = new C(); | Does Not Compile |
| (D)c.methodC(); | Does Not Compile |

}

****