Compile time binding to a method signature and run time method

For a method invocation   a1.method(list-of-params) :

1) At compile time the DECLARED classes of the variables which serve as parameters in the method invocation are used to choose the most specific method signature visible from the DECLARED class of variable a1 - the choice set includes all visible inherited methods. The return type is not considered at this stage.

2) At run time an exact match of the compile time descriptor (signature selected above plus return type) is sought in the Actual class associated with the object instance held in variable a1, and, if needed, then is sought in its parent and ancestor classes in order. The classes of the actual parameters are NOT considered at run time.

An example that might be surprising at first is:

```
  class A {
(0)   public void method1(A a1, A a2)...
      public void method1(A a1, int i)...
      }

  class B extends A {
(1)   public void method1(A a3, A a4)...
(2)   public void method1(B b1, B b2)...

      public static void main(String [] args) {
          A a6;
          B b3, b4, b5;
          a6 = b3;
(3)       a6.method1(b4, b5);   ....
      }
  }
```

The method invocation at (3) causes the method at (1) to be invoked at run time, even tho (2) is a more specific and matching method in the class of the invoking object instance (object b3).

This happens because compile time analysis selects the method signature (0) as the only visible applicable signature in the declared class (A) of variable a6, which is used to invoke the method. At run time, this signature is used to select (1) since it matches the selected compile-time descriptor (signature + return type).