CS 213 : Software Methodology

Spring 2017

Sesh Venugopal

Lecture 2: Jan 19
Static and Dynamic Types

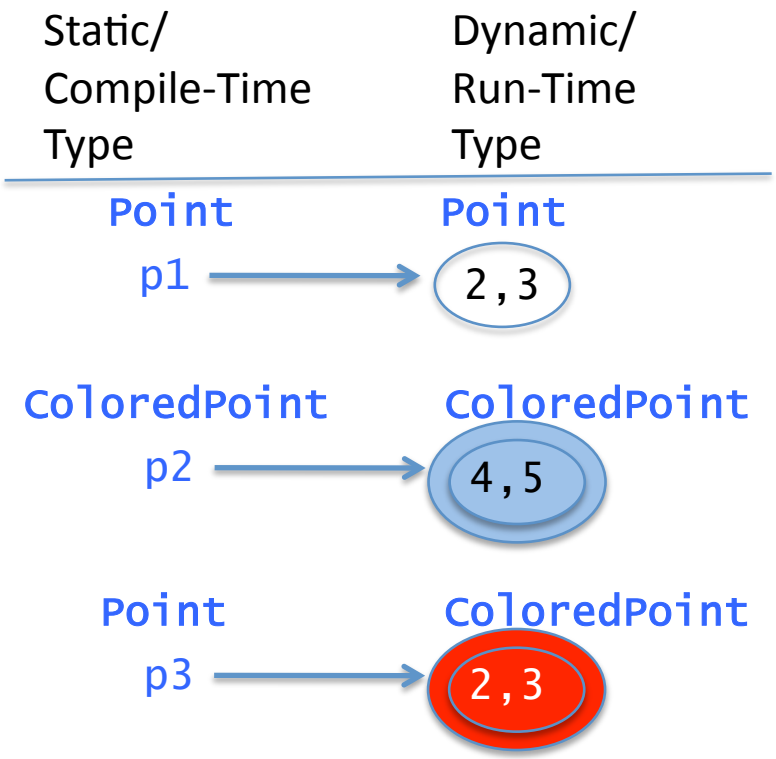
Static and Dynamic Types

```
public class PointApp {
   public static void
   main(String[] args) {

      Point p1 = new Point(2,3);


      ColoredPoint p2 =
        new ColoredPoint(4,5,"blue");


      Point p3 =
        new ColoredPoint(2,3,"red");
}
```



Every `ColoredPoint` is a `Point` (just like every Student is a Person) – so any `ColoredPoint` instance (dynamic type) can be referred to by a `Point` variable (static type)

Dynamic Binding

```
public class PointApp {
   public static void
   main(String[] args) {
      Point p1 = new Point(2,3);
      ColoredPoint p2 = new ColoredPoint(4,5,"blue");
      Point p3 = new ColoredPoint(2,3,"red");
      System.out.println(p2.getColor()); // ? "blue"
      System.out.println(p3.getX()); // ? 2
      System.out.println("p3 = " + p3); // ? "p3 = 2,3,red"
}
```

Dynamic Binding

Static type of p3 is `Point`, but dynamic type (type of instance it points to) is `ColoredPoint`.

So, the `p3.toString()` static call is bound to the dynamic type, `ColoredPoint`.

This results in the overridding version of `toString()` being executed.

Static and Dynamic Types

```
public class PointApp {
   public static void
   main(String[] args) {

      ColoredPoint p4 = new Point(5,6); // ?

}
```

WILL NOT COMPILE
Every `Point` (RHS) is
NOT a `ColoredPoint`
(LHS), so a `Point` instance
cannot be referenced
by a `ColoredPoint` variable

Static and Dynamic Types

```
public class PointApp {
   public static void
   main(String[] args)

      Point p5 = new ColoredPoint(1,2,green);

      System.out.println(p5.getColor());  // ?

}
```

WILL NOT COMPILE
Because the static type of `p5` is `Point`, ONLY members of `Point` class can be syntactically referenced by `p5`. Since `getColor` is not in the Point class, compiler flags error