CS 213 : Software Methodology
Spring 2023

Sesh Venugopal

Jan 30

Inheritance & Static Members

Inheritance - Static Fields

```
public class Supercl {
   static int x=2;
   public static void m() {
      System.out.println("in class Supercl");
   }
}
```

```
public class Subcl
extends Supercl {
   int x=3;
}
```

Instance field with same name as inherited static field x

```
public class StaticTest {
   public static void main(String[] args) {
      System.out.println(Subcl.x); // ? DOES NOT COMPILE
   }
}
```

"cannot make static reference to non-static field x"

Instance field of same name will HIDE inherited static field

Inheritance - Static Fields

```
public class Supercl {
   static int x=2;
   public static void m() {
      System.out.println("in class Supercl");
   }
}
```

What if we write a static method in Subcl to get at the inherited static x?

```
public class Subcl
extends Supercl {
   int x=3;
   public static int getX() {
      return x;
   }
}
```

WILL THIS COMPILE?

NO

"cannot make static reference to non-static field x" – same as before

Inheritance - Static Fields

```
public class Supercl {
   static int x=2;
   public static void m() {
      System.out.println("in class Supercl");
   }
}
```

```
public class Subcl
extends Supercl {
   int x=3;
}
```

```
public class StaticTest {
   public static void main(String[] args) {
      Subcl subcl  = new Subcl();
      System.out.println(subcl.x); // ? 3 – instance field x
      Supercl supercl  = new Subcl();
      System.out.println(supercl.x); // ? 2 – inherited static field x  ! ! !
   }
}
```

static type

dynamic type

INHERITED STATIC FIELDS ARE STATICALLY BOUND (TO REFERENCE TYPE), NOT DYNAMICALLY BOUND (TO INSTANCE TYPE)

Static Method Call Binding

```
public class Sorter {

    public static void
    sort(String[] names) {
        System.out.println(
          "simple sort";
        }
    }
}
```

```
public class IllustratedSorter
extends Sorter {

    // override
    public static void
    sort(String[] names)
        System.out.println(
          "illustrated sort";
        }
    }
}
```

```
Sorter p = new IllustratedSorter();
```

static type

dynamic type

```
p.sort(); // ? "simple sort"
```

`sort()` is statically bound to `p`, meaning since `Sorter` is the static type of `p`, the `sort()` method in `Sorter` is called

Overriding a static method with an instance method

```
public class Sorter {

    public static void
    sort(String[] names) {
        System.out.println(
          "simple sort";
        }
    }
}
```

```
public class IllustratedSorter
extends Sorter {

    // override
    public ~~static~~ void
    sort(String[] names)
        System.out.println(
          "illustrated sort";
        }
    }
}
```

COMPILE?

WILL NOT COMPILE: "Instance method cannot override static method sort from Sorter"

Overriding an instance method with a static method

```
public class Sorter {

    public static void
    sort(String[] names) {
        System.out.println(
          "simple sort";
        }
    }
}
```

```
public class IllustratedSorter
extends Sorter {

    // override
    public static void
    sort(String[] names)
        System.out.println(
          "illustrated sort";
        }
    }
}
```

COMPILE?

WILL NOT COMPILE: "Static method cannot override instance method sort from Sorter"