CS 213 : Software Methodology

Spring 2019

Sesh Venugopal

Lecture 3: Jan 29

Inheritance: Private Fields/Static Members

Inheritance - Private Fields

```
public class Point {
   private int x,y;
   ...
}


public class ColoredPoint extends Point {
   // x and y inherited but HIDDEN
   ...
   public int getX() { // override inherited getX()
      return x;
   }
}
```

COMPILE?

WILL NOT COMPILE
because x is hidden

Inheritance - Private Fields

```
public class Point {
   private int x,y;
   ...
}
```

```
public class ColoredPoint extends Point {
   // x and y inherited but HIDDEN
   ...  // getX() is NOT overridden
}
```

```
public class PointApp {
   public static void
   main(String[] args) {
      ColoredPoint cp = new ColoredPoint(4,5,"blue");

      System.out.println(cp.x); // ?
```

WILL NOT COMPILE, x is hidden

```
      System.out.println(cp.getX()); // ? 4
```

Inherited `getX()` method is able to access the `x` field

```
}
```

Inheritance - Static Members

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

```
public class Subcl
extends Supercl { }
```

```
public class StaticTest {
   public static void main(String[] args) {
      Supercl supercl = new Supercl();
      System.out.println(supercl.x);  // ?  0
      supercl.m(); // ?  "in class Supercl"
      Subcl subcl = new Subcl();
      System.out.println(subcl.x); // ?  0 – inherited from Supercl
      subcl.m(); // ? "in class Supercl"  – inherited from Supercl
   }
}
```

Inheritance - Static Fields

```
public class Supercl {
   static int x;
   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;
   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); // ? 0 – 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();

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

static type

dynamic type

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