Programming Languages and Techniques (CIS120)

Lecture 25

March 26, 2014

Immutable Lists in Java

Inner classes

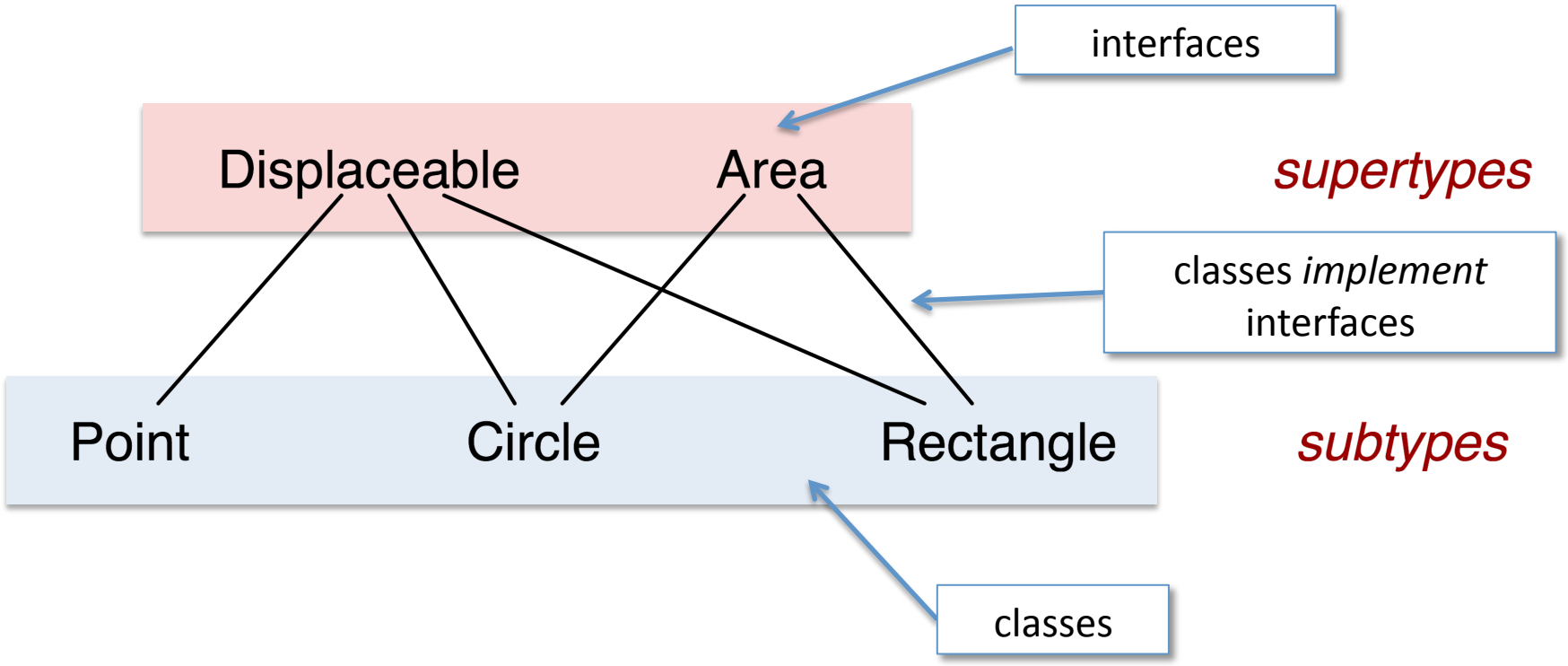
Announcements

- HW08 (GUI Programming II) is due next Tuesday at 11:59:59pm

- *Midterm 2 is Friday, April 4th in class*
  - Mutable state (in OCaml and Java)
  - Objects & encapsulation (in OCaml and Java)
  - ASM (in OCaml and Java)
  - Reactive programming (in OCaml and Java)
  - Arrays (in Java)
  - Subtyping & Inheritance (in Java)

Subtypes and Supertypes

- An interface represents a *point of view* about an object
- Classes can implement *multiple* interfaces



Types can have many different supertypes / subtypes

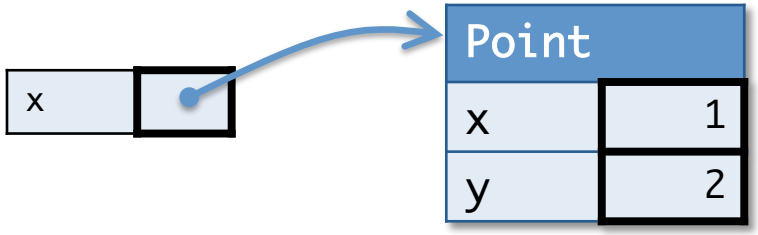
"Static" types vs. "Dynamic" classes

- The **static type** of an *expression* is a type that describes what we (and the compiler) know about the expression at compile-time (without thinking about the execution of the program)

```
Displaceable x;
```

- The **dynamic class** of an *object* is the class that it was constructed from at run time

```
x = new Point(1,2)
```



- Sometimes the static type of an expression is a class, sometimes it is an interface
  - The dynamic class will always be a *subtype* of its static type

Static type vs. Dynamic class quiz

```
public Area asArea (Area s) {
   return s;
}
…
Rectangle r =
  new Rectangle (1,2,1,1);
Circle c = new Circle (1,1,3);
Area s1 = r;  // A
Area s2 = c;  // B
s2 = r;       // C

__D__ x = asArea (r);
__E__ y = asArea (s1);

s1 = c;   // F
s1 = s2;  // G
r = c;    // H
r = s1;   // I
```

What is the static type of s1 on line A?

1. Rectangle
2. Circle
3. Area
4. none of the above

Static type vs. Dynamic class quiz

```
public Area asArea (Area s) {
   return s;
}
…
Rectangle r =
  new Rectangle (1,2,1,1);
Circle c = new Circle (1,1,3);
Area s1 = r;  // A
Area s2 = c;  // B
s2 = r;       // C

__D__ x = asArea (r);
__E__ y = asArea (s1);

s1 = c;   // F
s1 = s2;  // G
r = c;    // H
r = s1;   // I
```

What is the dynamic class of s1 when execution reaches A?

1. Rectangle
2. Circle
3. Area
4. none of the above

Static type vs. Dynamic class quiz

```
public Area asArea (Area s) {
   return s;
}
…
Rectangle r =
  new Rectangle (1,2,1,1);
Circle c = new Circle (1,1,3);
Area s1 = r;  // A
Area s2 = c;  // B
s2 = r;       // C

__D__ x = asArea (r);
__E__ y = asArea (s1);

s1 = c;   // F
s1 = s2;  // G
r = c;    // H
r = s1;   // I
```

What is the static type of s2 on line B?

1. Rectangle
2. Circle
3. Area
4. none of the above

Static type vs. Dynamic class quiz

```
public Area asArea (Area s) {
   return s;
}
…
Rectangle r =
  new Rectangle (1,2,1,1);
Circle c = new Circle (1,1,3);
Area s1 = r;  // A
Area s2 = c;  // B
s2 = r;       // C

__D__ x = asArea (r);
__E__ y = asArea (s1);

s1 = c;   // F
s1 = s2;  // G
r = c;    // H
r = s1;   // I
```

What type should we declare for x (in blank D)?

1. Rectangle
2. Circle
3. Area
4. none of the above

Static type vs. Dynamic class quiz

```
public Area asArea (Area s) {
   return s;
}
…
Rectangle r =
  new Rectangle (1,2,1,1);
Circle c = new Circle (1,1,3);
Area s1 = r;  // A
Area s2 = c;  // B
s2 = r;       // C

__D__ x = asArea (r);
__E__ y = asArea (s1);

s1 = c;   // F
s1 = s2;  // G
r = c;    // H
r = s1;   // I
```

What is the dynamic class of x?

1. Rectangle
2. Circle
3. Area
4. none of the above

Static type vs. Dynamic class quiz

```
public Area asArea (Area s) {
   return s;
}
…
Rectangle r =
  new Rectangle (1,2,1,1);
Circle c = new Circle (1,1,3);
Area s1 = r;  // A
Area s2 = c;  // B
s2 = r;       // C

__D__ x = asArea (r);
__E__ y = asArea (s1);

s1 = c;   // F
s1 = s2;  // G
r = c;    // H
r = s1;   // I
```

What type should we declare for y (in blank E)?

1. Rectangle
2. Circle
3. Area
4. none of the above

Static type vs. Dynamic class quiz

```
public Area asArea (Area s) {
   return s;
}
…
Rectangle r =
  new Rectangle (1,2,1,1);
Circle c = new Circle (1,1,3);
Area s1 = r;  // A
Area s2 = c;  // B
s2 = r;       // C

__D__ x = asArea (r);
__E__ y = asArea (s1);

s1 = c;   // F
s1 = s2;  // G
r = c;    // H
r = s1;   // I
```

What is the dynamic class of y?

1. Rectangle
2. Circle
3. Area
4. none of the above

Static type vs. Dynamic class quiz

```
public Area asArea (Area s) {
   return s;
}
…
Rectangle r =
  new Rectangle (1,2,1,1);
Circle c = new Circle (1,1,3);
Area s1 = r;  // A
Area s2 = c;  // B
s2 = r;       // C

__D__ x = asArea (r);
__E__ y = asArea (s1);

s1 = c;   // F
s1 = s2;  // G
r = c;    // H
r = s1;   // I
```

Is the assignment on line F well typed?

1. yes
2. no

Static type vs. Dynamic class quiz

```
public Area asArea (Area s) {
   return s;
}
…
Rectangle r =
  new Rectangle (1,2,1,1);
Circle c = new Circle (1,1,3);
Area s1 = r;  // A
Area s2 = c;  // B
s2 = r;       // C

__D__ x = asArea (r);
__E__ y = asArea (s1);

s1 = c;   // F
s1 = s2;  // G
r = c;    // H
r = s1;   // I
```

Is the assignment on line G well typed?

1. yes
2. no

Static type vs. Dynamic class quiz

```
public Area asArea (Area s) {
   return s;
}
…
Rectangle r =
  new Rectangle (1,2,1,1);
Circle c = new Circle (1,1,3);
Area s1 = r;  // A
Area s2 = c;  // B
s2 = r;       // C

__D__ x = asArea (r);
__E__ y = asArea (s1);

s1 = c;   // F
s1 = s2;  // G
r = c;    // H
r = s1;   // I
```

Is the assignment on line H well typed?

1. yes
2. no

Static type vs. Dynamic class quiz

```
public Area asArea (Area s) {
   return s;
}
…
Rectangle r =
  new Rectangle (1,2,1,1);
Circle c = new Circle (1,1,3);
Area s1 = r;  // A
Area s2 = c;  // B
s2 = r;       // C

__D__ x = asArea (r);
__E__ y = asArea (s1);

s1 = c;   // F
s1 = s2;  // G
r = c;    // H
r = s1;   // I
```

Is the assignment on line I well typed?

1. yes
2. no

Datatypes in Java

Immutable Lists in Java

Datatypes and lists in Java

What is the Java analogue of OCaml (immutable) lists…

```
type string_list = Nil | Cons of string * string_list
```

…and recursive/iterative functions over lists?

```
let rec number_of_songs (pl : string_list) : int =
   begin match pl with
    | [] -> 0
    | ( song :: rest ) -> 1 + number_of_songs rest
   end
```

Datatypes and Immutable Lists in Java

```
interface StringList {
   public int numberOfSongs();
}
```

```
class Cons implements StringList {

   private final String head;
   private final StringList tail;

   public Cons (String h,
                StringList t) {
      head = h; tail = t;
   }

   public int numberOfSongs() {
       return
          1 + tail.numberOfSongs();
   }
}
```

```
class Nil implements StringList {









   public int numberOfSongs() {
     return 0;
   }
}
```

Datatypes and Immutable Lists in Java

```
interface StringList {
   public boolean isNil();
   public String hd();
   public StringList tl();
}
```

only call these if isNil() == false

```
class Cons implements StringList {
   private final String head;
   private final StringList tail;
   public Cons (String h,
                StringList t) {
      head = h; tail = t;
   }
   public boolean isNil() {
     return false;
   }
   public String hd() {
     return head;
   }
   public StringList tl() {
     return tail;
   }
}
```

```
class Nil implements StringList {

   public boolean isNil() {
     return true;
   }
   public String hd() {
     return null;
   }
   public StringList tl() {
     return null;
   }
}
```

Creating lists

OCaml

```
let x = Cons “Dark Horse” (Cons “Roar” Nil)
```

Java

```
StringList x = new Cons (“Dark Horse”, new Cons (“Roar”, new Nil()))
```

- Both lists are immutable:
  - In Java, can’t say `x.tail = new Nil()`
  - tail defined as `final`

- General pattern for datatypes:
  - Define an interface for the datatype type
  - For each data constructor, add a class
  - Add accessors for data (clunky, we'll see better ways to do this later)

Using lists

OCaml

```
let rec number_of_songs (pl : string_list) : int =
   begin match pl with
    | [] -> 0
    | ( song :: rest ) -> 1 + number_of_songs rest
   end
```

Java

```
public static int numberOfSongs (StringList pl) {
   if (pl.isNil()) {
      return 0;
   } else {
      return 1 + numberOfSongs (pl.tl());
   }
}
```

```
interface StringList {
   public boolean isNil();
   public String hd();
   public StringList tl();
}
```

List Iteration

OCaml (Better)

```
let number_of_songs (pl : string_list) : int =
  let rec loop (pl:string list) (n:int) : int =
    begin match pl with
    | [] -> n
    | ( song :: rest ) -> number_of_songs rest (1 + n)
    end
  in loop pl
```

Java (Better)

```
public static int numberOfSongs (StringList pl) {
   int n = 0;
   StringList curr = pl;
   while (!curr.isNil()) {
      n = 1 + n;
      curr  = curr.tl();
   }
   return n;
}
```

no tail recursion in Java

Using mutable local variables for value-oriented programming

First-class functions?

OCaml

```
let rec transform (f  : string -> string)
             (pl : string_list) : string_list =
   begin match pl with
    | [] -> []
    | ( song :: rest ) -> f song :: transform f rest
   end

let y = transform String.uppercase (Cons “dynamite” Nil)
```

Java

```
public static StringList transform (??? f, StringList pl) {
   if (pl.isNil()) {
      return new Nil();
   } else {
      return new Cons (????, transform(f, pl.tail()));
   }
}

public static void testTransform() {
   StringList x = new Cons (“dynamite”, new Nil());
   StringList y = transform(???, x);
   assertEqual (y.hd(), “DYNAMITE”);
}
```

First-class functions via objects

```
public interface Fun {
   public String apply (String x);
}
```

```
public class UpperCaseFun implements Fun {
   public String apply (String x) {
      return x.toUpperCase();
   }
}
```

```
 public static StringList transform (Fun f, StringList pl) {
   if (pl.isNil()) {
      return new Nil();
   } else {
      return new Cons (f.apply(pl.hd()), transform(f, pl.tl()));
   }
 }

 public static void testTransform() {
   StringList x = new Cons ("dynamite", new Nil());
   StringList y = transform (new UpperCaseFun(), x);
   assertEqual (y.hd(), "DYNAMITE");
 }
```

First-class functions via inner classes

```
public interface Fun {
   public String apply (String x);
}
```

```
public static StringList transform (Fun f, StringList pl) {
   if (pl.isNil()) {
     return new Nil();
   } else {
     return new Cons (f.apply(pl.hd()), transform(f, pl.tl()));
   }
}
```

```
StringList z = transform
                    (new Fun() {
                      public String apply (String x) {
                        return x.toUpperCase();
                      }},
                     x);
```