cs2220: Engineering Software

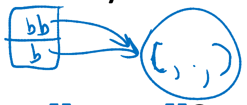
Class 14: Object-Oriented Programming

Fall 2010
University of Virginia
David Evans

Menu

- Subtyping with Parameterized Types
  - Arrays
  - Generics
- "Object-Oriented Programming"

Subtyping and Arrays



Does **B** $\subseteq$ **A** imply **B[]** $\subseteq$ **A []**?

BlackBear ≤ Bear

BlackBear [] ≤ Bear[]
?

GrizzlyBear gb = ...
Bear [] b = ... ;
b[0] = gb;

BlackBear [] bb = ... ;
~~bb = b;~~
b = bb;
b[0] = new Grizzly();

Array Subtyping

```
static public Object getFirst (Object [] els) throws NoSuchElementException {
   if (els == null || els.length == 0) {
      throw new NoSuchElementException ();
   } else {
      return els[0];
   }
}

static public void main (String args[]) {
    try {
        Object o = getFirst (args);
        System.err.println ("The first parameter is: " + o);
    } catch (NoSuchElementException e) {
        System.err.println ("There are no parameters!");
    }
}
```

Array Store

```
static public void setFirst (Object [] els) throws NoSuchElementException {
   if (els == null || els.length == 0) {
      throw new NoSuchElementException ();
   } else {
      els[0] = new Object ();
   }
}
```

```
> javac TestArrays.java
> java TestArrays test
The first parameter is: test
Exception in thread "main" java.lang.ArrayStoreException
      at TestArrays.setFirst(TestArrays.java:16)
      at TestArrays.main(TestArrays.java:25)
```

```
static public void main (String args[]) {
    try {
        Object o = getFirst (args);
        System.err.println ("The first parameter is: " + o);
        setFirst (args);
    } catch (NoSuchElementException e) {
        System.err.println ("There are no parameters!");
    }
}
```

Java's Array Subtyping Rule

Static type checking: B <= A $\Rightarrow$ B[] <= A[]

Need a **run-time check** for every array store to an array where the actual element type is not known

What would be a better rule?

B ≤ A $\not\Rightarrow$ B[] ≤ A[]

Generic Subtyping

Does **B** $\subseteq$ **A** imply **T<B>** $\subseteq$ **T<A>**?

$\text{List}\langle\text{String}\rangle \subseteq \text{List}\langle\text{Object}\rangle$

$\cancel{?}$

Generic Subtyping: Novariant

```
List<String> as;
List<Object> ao;

ao = as;   Type mismatch: cannot convert from List<String> to List<Object>
as = ao;   Type mismatch: cannot convert from List<Object> to List<String>
```

Wildcard Types!

```
List<? extends Object> ag;

ag = ao;
ag = as;

String s = ag.get(0);
Type mismatch: cannot convert from capture#3-of ? extends Object to String
```

List<String> as;

:

String s = as.get(0);

from Class 2...

Buzzword Description

"A ~~simple~~, **object-oriented**, distributed, interpreted, **robust**, **secure**, architecture neutral, portable, high-performance, **multithreaded**, and dynamic language." [Sun95]

As the course proceeds, we will discuss how well it satisfies these "buzzwords". You should especially be able to answer how well it satisfies each of the blue ones in your final interview.