Section 7!

Concerning such fascinating topics as generics, switch statements, and homework!

With material borrowed from Marty Stepp, David Notkin, and Joshua Block (Effective Java ch. 5)

CSE 331, 11/10/11 *TA: Krysta Yousoufian*

Homework

- A4 due yesterday…
- Feedback on A2, A4? (new this quarter)
- A2 examples

Switch Statement

- Continued from last time
- www.cs.washington.edu/education/courses/cse331/11au/sections/zoo/Zoo.java
- Have finite set of values (enum or otherwise)
- Want to check which value a variable matches and do something
- Example: user enters menu option
- "Syntactic sugar": anything you do with a switch, you can do with if/else (but it's not as pretty)

Generics

Optional reading: Effective Java ch. 5

Demos

- Download all demos here
  - Note: demo for switch is also in Generics package
- Generic types:
  - List.java
  - ArrayList.java
- Generic methods:
  - SetUtils.java (method definitions)
  - GenericsTest.java (calls methods in SetUtils)
- (Possibly) more concrete example: Zoo
  - Exhibit.java (exhibits of animals in a zoo)
  - ZooMain.java (constructs a bunch of exhibits)

Generic Types

- Each object maps to some unknown type
- Set<Integer>, Set<String>, Set<Object>, …
- Coding example: List<E>

Generic Methods

- Use generics without creating a generic type
- *Method* uses some unknown type (parameters, maybe return value)
- To declare a method generic, put <E> (or <T> or …) before the return type

```
public static <E> void add(Set<E> items, E element)
public static <T> Set<T> union(Set<T> s1, Set<T> s2)
```

- When calling, you might need to specify the generic type…

```
Foo.<String>add(s1, "Hello world");
```

- … But usually the compiler can figure it out

```
Foo.<String>add(s1, "Hello world");
```

- Example: SetUtils.union()

Wildcards

- You have an object of a generic type, but you don't care what its type parameter is
  - You care that you have a Set
  - You don't care if it's a Set<Integer>, Set<String>, …
- Usage:
  - Use <?> instead of <E>
  - Why not use raw type ("Set") instead of wildcard ("Set<?>")?
  - (Almost) never use raw types – not type-safe
- Read "Set<?>" as "Set of some type"
- Example: intersectionCount()

Wildcards

When **not** to use

- Adding items – can't add anything but null to Collection<?>
- If type param appears more than once in method declaration
- In a return type – caller has to handle this specially
- Anytime you need to refer to the type (e.g. creating a new Collection<E>)
- Example: union() – creates new Set<E>
- Example: addAll() – adding items

Bounded Wildcards

- Extends
  - Set<? extends Foo>
  - Requires Foo or a subtype of Foo
  - e.g. Set<? extends Number> will accept Set<Integer>
  - Example: unionBetter()
- Super
  - Set<? Super Foo>
  - Requires Foo or a supertype of Foo
  - e.g. Set<? Super Integer> will accept Set<Number>
  - Example: addAllBetter()