

Java 1.5 Features (I)

Foreach and enums

Administrivia

- Labs
 - Took email feedback
 - Consensus: Labs are valuable, but take a lot of time
 - Result: We will continue to have lab material, but will work to ensure that they are completable during lab period
- Questions on P3?

New in Java 1.5

- Originally, I intended to do about half this class in Java 1.5
- Support not yet widespread enough (grrrrrr)
 - No Eclipse
 - No Apple support (\Rightarrow no live demo code)
- Some of you have discovered a lot of neat 1.5 tricks already anyway...

New in Java 1.5

- Many (semi-)useful tricks
 - Static import
 - Printf
 - Variable length arg lists
 - Foreach (sort-of)
 - Enums
 - Parameterized types (“generics”)

Static import

- Lets you import just the static components of a class as top-level names in your code
- E.g., `out.println()` rather than `System.out.println()`
- Cuts down typing
- 2 really significant uses (IMNSHO)
- `import static Math.*;`
 - Math full of static functions (`cos()`, `abs()`, etc.)
`double d=cos(PI/7);`
 - Instead of
`double d=Math.cos(Math.PI/7); // ick...`

2nd important use

```
public class BigHairyClassName extends Lexer {  
    public static final int TT_NAME=0;  
    public static final int TT_VERB=1;  
    public static final int TT_FUNCTION=2;  
    ...  
}
```

```
// elsewhere
```

```
if (t.getType()==BigHairyClassName.TT_FUNCTION)
```

```
// my fingers are now falling off...
```

The joy of printf

- In Java 1.4, printing out floating point numbers basically... Sucked.
- Default:
`System.out.println(Math.PI)` => 3.141592653589793
- Could get limited precision/formatted floating point numbers.
- That sucked too.
- Involved extra formatter objects.
- Nicely OO and decomposed.
- Usability basically blew chunks.

printf

- “Print formatted”
- Relic of the C (or pre-C) days
- New and improved for Java 1.5
- Still looks basically the same, though:

```
out.printf("%.3f\n", PI)
```
- Member of `java.io.PrintWriter/PrintStream`
- Still just as arcane
 - Have to memorize/look up all formatting codes
 - Java can't check format string at compile time

```
out.printf("%.3f\n", "omphaloskepsis")
```
 - **Oops.** `IllegalFormatException`

Variable length arg lists

- Check out the signature for printf:
 - `public PrintWriter format(String format, Object... args)`
- Those ellipses aren't laziness -- they're really part of the object signature
- Tells Java: "I don't know how many args there will be to this function, so just let the programmer enter as many as he/she wants to"
- Here `args` has type `Object[]`
- Java handles creating the array so user doesn't see it
- Code inside method sees full array, though

Enumerations

- Often (*very often*) you need to be able to specify a small, finite set
 - `SMALL, MEDIUM, LARGE`
 - `HEARTS, SPADES, DIAMONDS, CLUBS`
 - `WORD, NUMBER, AND, OR, LPAREN, RPAREN, ...`
- Can specify these with `constant (static final) ints`
- Problems:
 - Can't type-check
 - Litter code w/ lots of cases
 - Hard to extend later

New enum mechanism

- 1.5 introduces enums
- Basic use simple:

```
public enum Fruit { APPLE, BANANA, ORANGE,  
KUMQUAT }
```

```
Fruit f=Fruit.APPLE;
```

```
if (f==Fruit.BANANA) { // do something }
```

```
public void _processFruit(Fruit f) { ... }
```

- Provides a small, fixed set, plus type safety

Stupid fruit tricks

- Enums are actually classes
- Each instance of the enum is a separate class
 - Fruit.APPLE is its own class
- Members of that class are *singletons* -- only ever one of them
 - Only one APPLE in the whole program (per JVM)
- Also, b/c they're full-fledged classes, you can give them constructors, extra data elements, etc.