Programming Languages and Techniques (CIS120e)

Lecture 24
Nov. 5, 2010

Histogram Example Cont’d, Exceptions
Announcements

• Homework 8 (SpellChecker) is due ???.

• Midterm 2 is Friday, November 12th
Design Example: Histogram.java

A design exercise using java.io and generic collections libraries.
Java Pragmatics Cheat Sheet

• Program entry point: `public static void main(String[] args)`
  – Command-line arguments are passed in the String array given to main.

• Generic types cannot be instantiated by primitive datatypes (e.g. int, boolean); instead you must use “wrapper” classes (e.g. Integer, Boolean)
  – Java will automatically convert primitive values to wrapped objects.
  – See `java.lang.Integer`, `java.lang.Character`
  – This is a “kludge” due to Java’s history; generics weren’t added until long after the Java virtual machine was standardized…

• When creating an object of generic type, don’t forget to give type parameters: e.g. `new TreeMap<String, Integer>()`
Java Pragmatics Cheat Sheet

• Static fields and methods are “global” variables attached to a class name.
  – e.g. Character.isLetter(int c)

• Classes can be nested: e.g. Map.Entry<K,V>

• Abstract classes can’t be instantiated, but they make good types.
  – Libraries use abstract classes to encapsulate shared algorithms.

• Calls to overloaded methods and constructors are determined by the number of arguments and their static types.

• Many I/O methods can fail by throwing an exception.
  – Exceptions are for unusual situations: File does not exist, Disk is full, etc.
  – Code that calls such methods can handle the error using:
    
    ```java
    try {...} catch (Exception e) {...}
    ```
Exceptions

Dealing with failure.
Sources of Failure

- Some methods might require that their arguments satisfy certain *preconditions*.
  - e.g. the input to `max` is a non-empty list
- External components might fail.
  - e.g. you ask the OS to open a file that does not exist
- Resources might be exhausted
  - e.g. your program uses all of the computer’s memory
- The implementation might not be complete
  - e.g. some Iterators don’t support the “remove” operation

- These are all *exceptional circumstances*...
  - how do we deal with them?
Ways to Handle Failure

• Return an error or default value.
  – e.g. `read` returns –1 if no more input is available.
  – `Math.sqrt` returns `NaN` (“not a number”) if given an input < 0.
  – Many Java library routines return `null`.
  – Use with caution – easy to introduce hard to find bugs because it
    passes responsibility to the caller.
  – Especially bad if the answer can be interpreted meaningfully

• Use an informative datatype
  – e.g. in OCaml we used ‘a option to signal partiality
  – also passes responsibility to the caller, but the compiler forces the
    caller to do the proper check

• Use exceptions.
  – in OCaml we used “failwith”
  – in Java we can throw an exception.
Exceptions

• An exception is an object representing abnormal conditions.
  – Its internal state describes what went wrong.
  – Examples: NullPointerException, IllegalArgumentException, IOException

• **Throwing** an exception is an *emergency exit* from the current method.
  – The exception propagates up the invocation stack until it either reaches the top and the stack, in which case the program aborts with the error, or the exception is caught.

• **Catching** an exception lets callers take appropriate actions to handle the abnormal circumstances.