

Agenda: Discussion Week 5

April 27, 2009

- Always deliverable
- Exceptions
- Assignment and comparison
- Searching in a list
- Generics
- PSA 3
- Suggestions?

Goal: Always Deliverable

- Code compiles and program works
 - The part that is done works
- Coding ideas
 - Start with comments, method stubs (outline)
 - Add code in small increments, compile and test frequently
- Debugging ideas
 - At compile time:
 - Look at **1st** compiler error (description, line number)
 - Comment out code (locate the problem)
 - At runtime, use print statements (or java debugger)
 - Trace program execution
 - Trace values of variables

Exceptions (1)

- Getting more details
 - Print short description of exception
 - `String toString()`
 - Print stack trace
 - `void printStackTrace()`
 - Look at 1st line of trace (description, line number)
 - Lookup error message
 - Google
- References
 - Tutorial
 - <http://java.sun.com/docs/books/tutorial/essential/exceptions>
 - API
 - <http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Exception.html>

Exceptions (2)

```
System.out.println("Before try");
try {
    System.out.println("Throwing Exception");
    throw new Exception("This is our Exception");
} catch(Exception ex) {
    System.out.println("Caught Exception");
    ex.printStackTrace();
} finally {
    System.out.println("Finally");
}
System.out.println("After catch");
```

Assignment and comparison

- =
 - Assignment operator
 - Assigns the value on the right to the operand on the left
 - When used on primitives, assigns primitive value
 - When used on reference types (objects, arrays), assigns reference
- == and !=
 - Comparison operators (equal, not equal)
 - Compares references (not values)
 - Examples: null reference? references point to same object?
- equals
 - Method in Object class: boolean equals(Object obj)
 - Sub-classes may implement their own equals methods
 - Compares values for equality
 - Example: Do objects have same value?

Searching a list for some element

- Search in an ArrayList or Array
- Pattern A: loop over everything
 - break out of loop, then return after loop
 - OR --
 - return inside of loop
- Pattern B: loop until found or reached end of list
 - return after loop

Search pattern A

- Pattern A: loop over everything
 - break out of loop, then return after loop
 - OR --
 - return inside of loop

```
ArrayList list = {"A", "B", "C"};
String letter = "C";
int index = -1;
for(int i=0; i < list.size(); i++) {
    if (list.get(i).equals(letter)) {
        index = i;
        break;
    }
}
return index;
```

Search pattern B

- Pattern B: loop until found or reached end of list
 - return after loop

```
boolean found=false;
ArrayList list = {"A", "B", "C"};
String letter = "C";
int i = 0;
int index = -1;
while((!found) && (i < list.size())) {
    if (list.get(i).equals(letter)) {
        found = true;
        index = i;
    }
    i++;
}
return index;
```

Generics

- Communicate the type of a collection to the compiler, so that it can be checked.
- <http://java.sun.com/j2se/1.5.0/docs/guide/language/generics.html>

Which of the following are legal instantiations of ArrayList?

1. `ArrayList<int> foo = new ArrayList<int>();`
2. `ArrayList<Double> foo = new ArrayList<Double>(200);`
3. `ArrayList<CreditCard> foo = new ArrayList<CreditCard>();`

PSA 3

- Abstract Data Types (ADTs)
 - provide a useful service, while hiding the implementation details (“black box”)
 - you interact with an ADT through set of well-defined methods (“API”)
- PSA 3 Classes
 - SmartArray
 - implement an ADT like ArrayList
 - SensorData, SensorReader
 - re-write the sample solutions to use SmartArray instead of ArrayList

PSA 3 (2)

- Code reuse
 - Lazy = good!
- ArrayList
 - <http://java.sun.com/j2se/1.5.0/docs/api/java/util/ArrayList.html>