Java Language Constructs II

Department of Computer Science
University of Maryland, College Park
Regarding Questions over E-mail

- Due to the large number of students in class, we (instructors and TAs) cannot address project questions, questions about lecture material, etc. over e-mail. If you have any questions, please address them in lab/discussion session, in lecture or during office hours. Thank you for your cooperation.
About Terpconnect Account

• Your first project requires a terpconnect account and this may take at least 24 hours. No project extensions will be granted due to terpconnect account problems. It is your responsibility to start the project ahead of time.
About Quizzes/Exams

• Please read the guidelines at:
About Style

• Let’s go over the following information
Implementing Equals

• Approach we want to use (assuming class A)

```java
generic boolean equals(Object obj) {
    if (obj == this)
        return true;
    if (!((obj instanceof A)) // covers obj == null case
        return false;
    A a = (A)obj;
    /* Specific comparison based on A fields appears here */
}
```

• What happens if we use comparisons of Class objects rather than instanceof?

• Example: equalsMethod package
Comparable Interface

- Comparable
  - public int compareTo(T o)
  - a.compareTo(b) returns
    - Negative if a < b, 0 if a == b, positive if a > b
- Properties
  - Referred to as the class's *natural ordering*
  - Can sort using Collections.sort() & Arrays.sort()
    - Example: Collections.sort(myList);
  - Can use as keys in SortedMap & SortedSet
  - Consistency w/ equals() strongly recommended
    - x.equals(y) if and only if x.compareTo(y) == 0
  - Example: comparableExample package
Comparator Interface

- Comparator
  - public int compare(T a, T b)
    - **Negative** if a < b, 0 if a == b, **positive** if a > b
- Properties
  - Imposes total ordering on objects of a class
  - Provide alternatives to natural ordering
  - Supports generics
    - Example: `class myC implements Comparator<Foo>{ … }`
  - Use as parameter for sort function
    - Example: `Collections.sort(myFooList, new myC( ));`
- Example: `comparatorExample`
Three Levels of Copying Objects

Assume y refers to object z

1. Reference copy
   - Makes copy of reference
   - \( x = y; \)

2. Shallow copy
   - Makes copy of object
   - \( x = y.clone(); \)

3. Deep copy
   - Makes copy of object z and all objects (directly or indirectly) referred to by z
Cloning

- Cloning
  - Creates identical copy of object using clone( )
- Cloneable interface
  - Supports clone( ) method
  - Returns copy of object
    - Copies all of its fields
    - Does not clone its fields
    - Makes a shallow copy
- Example: cloning package
Garbage Collection

• Concepts
  • All interactions with objects occur through reference variables
  • If no reference to object exists, object becomes garbage (useless, no longer affects program)

• Garbage collection
  • Reclaiming memory used by unreferenced objects
  • Periodically performed by Java
  • Not guaranteed to occur
  • Only needed if running low on memory
Destructor

• Description
  • Method with name `finalize()`
  • Returns void
  • Contains action performed when object is freed
  • Invoked automatically by garbage collector
    • Not invoked if garbage collection does not occur
  • Usually needed only for non-Java methods

• Example
  
  ```java
  class Foo {
      void finalize() { … } // destructor for foo
  }
  ```
Initialization Block

- Definition
  - Block of code used to initialize static & instance variables for class

- Motivation
  - Enable complex initializations for static variables
    - Control flow
    - Exceptions
  - Share code between multiple constructors for same class
Initialization Block Types

- Static initialization block
  - Code executed when class loaded
- Initialization block
  - Code executed when each object created
  - (at beginning of call to constructor)
- Example

```java
class Foo {
    static {
        A = 1;
    } // static initialization block
    {
        A = 2;
    } // initialization block
}
```
Variable Initialization

- Variables may be initialized
  - At time of declaration
  - In initialization block
  - In constructor

- Order of initialization
  1. Declaration, initialization block
     (in the same order as in the class definition)
  2. Constructor
Variable Initialization – Example

class Foo {
    static { A = 1; } // static initialization block
    static int A = 2; // static variable declaration
    static { A = 3; } // static initialization block
    { B = 4; } // initialization block
    private int B = 5; // instance variable declaration
    { B = 6; } // initialization block
    Foo() {
        A = 7;
        B = 8;
    } // constructor
} // now A = 7, B = 8 // initializations executed in order of number
Annotations

• Annotation → Java construct that allow us to add validity constraints to Java Classes

• Validity constraint example
  • A instance variable cannot assume a negative value
  • A parameter can not be null
  • A method in a class must override a method in its superclass

• Syntax
  at-sign (@) followed by annotation type and a parenthesized list of element-value pairs

• Example
  @DefaultAnnotationForParameters(NonNull.class)

• You can ignore annotations in code distribution for class projects