CSE332: Data Abstractions
Section 1

Nicholas Shahan
Winter 2015

Adapted from slides by Hye In Kim
Today

• Announcements
• Introductions
• Questions?
• Generics
• Project 1: Sound Blaster!
• Bugs & Testing
• Eclipse Tutorial
Announcements

• Project 1 is out
  – Phase A: Due Monday Jan 12\textsuperscript{th} 11pm
  – Phase B: Due Thursday Jan 15\textsuperscript{th} 11pm

• Written HW 1 is out
  – Due Friday Jan 16\textsuperscript{th}
Introductions
Your TAs & Office Hours

• Nicholas Shahan: Mon at 10:30 in CSE 218
• Jack Warren: Tues at 3:30pm in CSE 220
• Conrad Nied: Wed at 10:30 in CSE 218
• Daphna Khen: Wed at 3:30 in CSE 218
• Matthew Gillette: Thurs at 3:30 in CSE 220
• Ian Turner: Fri at 2:30 in CSE 218
• cse332-staff@cs.washington.edu
Introductions – Now You

• Name
• Year
• What is a movie that stands out in your mind? (for any reason good or bad)
Questions

Any questions about the class so far?
Generics

• Technique of writing Class/Interface without specifying type of data it uses
• The data types will be specified when the Class/Interface is used
• Idea: class/interface can have type parameter
  – Typically denoted as T or E
Generics Example

• Want a Bag class to store items

```java
public class Bag {  // Stores a String
    private String item;
    public void setItem(String x) { item = x; }
    public String getItem() { return item; }
}
```

```java
public class Bag {  // Stores a Book
    private Book item;
    public void setItem(Book x) { item = x; }
    public Book getItem() { return item; }
}
```

• What is the problem here?
Generics Example

- Should not create a Bag class for every type
- Pre Java 5: Objects – DO NOT USE

```java
public class Bag {
    private Object item;
    public void setItem(Object x) { item = x; }
    public Object getItem() { return item; }
}

Bag b = new Bag();
b.setItem("String in the bag.");
String contents = (String) b.getItem();
```

- What is the problem here?
Generics Example

• Pre Java 5: Objects – DO NOT USE

```java
Bag b = new Bag();  // Object Bag
b.setItem("String in the bag.");

String contents = (String) b.getItem();  // Ok
double contents = (double) b.getItem();  // Error (Runtime)
```

• Error appears at runtime

• Likely to crash 😞
Generics Example

• Should not create a Bag class for every type
• Using Java Generics

```java
public class Bag<E> {
    private E item;
    public void setItem(E x) { item = x; }
    public E getItem() { return item; }
}

Bag<String> b = new Bag<String>();
b.setItem("String in the bag.");
String contents = b.getItem();
```

• Why is this better?
Generics Example

• Should not create a Bag class for every type

• Using Java Generics

```java
Bag<String> b = new Bag<String>(); // Generic Bag
b.setItem("String in the bag.");

String contents = (String) b.getItem(); // Ok
double contents = (double) b.getItem(); // Error (Compile time)
```

• Error appears at compile time 😊
Generics

• Why Generics?
  – Type Safe Containers
  – Compile-time type checking
  – No need to cast or manually check the type

• Important: Cannot create a generic array!

E[] myArray = new E[INITIAL_SIZE];  // Error

@SuppressWarnings("unchecked")
E[] myArray = (E[]) new Object[INITIAL_SIZE];  // Ok
Project 1: Sound Blaster!

• Phase A
  – Implement Stack ADT: Stores double
    • Implement Dstack
    • Using Array (ArrayStack)
    • Using Linked List (ListStack)

• Phase B
  – Implement Stack ADT: Use generic
    • Implement GStack
    • Using Array (GArrayStack)
    • Using Linked List (GListStack)
Project 1: Sound Blaster!

• Reverse.java
  – Handles all music stuff
  – No need to edit for part A
  – Reverses in.dat file and writes it to out.dat
  – Accepts 4 command line arguments
    • Stack Implementation: array or list
    • Content type: double or generic
    • Input file name: (example) in.dat
    • Output file name: (example) out.dat

• Eclipse: Run with Command line arguments
Project 1: Sound Blaster!

• Sound Exchange (SOX)
  – Converts .wav file to .dat file & vice versa
    • Reverse.java needs .dat file
    • You need .wav file to play sound
  – Installed on lab machines
  – Use in terminal or command prompt

• Example Usage:
$> sox secret.wav secret.dat
Style Guide

• Style Points are up to 1/3 of your grade!!
  – Grade breakdown “roughly”:
    • 1/3 correctness
    • 1/3 write up
    • 1/3 style

• Make sure you read style guides
  – 142/143 Style tips:
  – 142/143 Unofficial Style Guide:
  – CSE 142/143 Unofficial commenting guide:

• We DO take points off for style!
Style Guide

• Make sure your code compiles without warnings
  – No correctness points if your code doesn’t compile!!
  – Use default package or remove package statement

• Comment your code

• Follow Java convention: this.isEmpty()  // smirk

• Use descriptive variable and method names
  – If variable points to beginning of queue, name it
    something like ‘front’ or ‘start’, not ‘w’ or ‘g’

• Use visibility specifiers (private/public etc.)
  – On every classes, methods, fields. Do not omit these!
Style Guide

• Initialize all non-static fields in constructor
• Make your code as concise and clear
• Use `@Override` when overriding
• Do not leave any warning-generating code
  – Suppress warnings only when you know exactly why it is there and why it is unavoidable
  – Suppress warnings on method/variable, but not on whole class
Style Guide

• Use constants instead of “magic numbers”
• Practice the art of “Boolean Zen”
• Maximize code reuse to minimize redundancy
  – For example: Re-use methods like `isEmpty()` instead of directly testing
  – also improves readability
• Note: a good compiler/run-time will in-line short methods so there is no loss in efficiency in doing this and it makes the code more readable.
• If any of these don’t make sense please ask!
Bugs & Testing

• Why Testing?
• Bugs can be costly
  – Cost points in homework
  – Can cost $$$ and even life (Therac-25)
• Interesting Bug References
  – List of bugs
  – History’s worst
  – Bugs of the month
Bugs & Testing

• Tips for Testing
  – Make sure program meets the spec
  – Test if each method works independently
  – Test if methods work together
  – Test for edge cases
    • Empty stack
    • Push after resizing
    • Anything else?
  – Check against Java’s Stack
Bugs & Testing

• Testing tools: JUnit Testing
  – Not required for Project 1
  – Required for Project 2

• Covered in a later section
  – If you are curious now, lets talk during office hours
Eclipse Tutorial

Select a workspace
Eclipse Tutorial

Create a new project
Eclipse Tutorial

Select the new project options
Eclipse Tutorial

Create a new class
Eclipse Tutorial

Run Configurations (command line arguments)
Eclipse Tutorial

Run Configurations (command line arguments)
Eclipse Tutorial

Run Configurations (command line arguments)
Eclipse Tutorial

• More Tutorials
  – Written Tutorial
    http://www.vogella.com/articles/Eclipse/article.html
  – Video Tutorial
    http://ecliptutorial.sourceforge.net/totalbeginner.html
  – Eclipse Shortcut Keys