Structural Programming and Data Structures

Winter 2000

CMPUT 102: Tracing Programs

Dr. Osmar R. Zaïane

University of Alberta

Course Content

- Introduction
- Objects
- Methods
- Tracing Programs
  - Object State
  - Sharing resources
  - Selection
  - Repetition
- Vectors
- Testing/Debugging
- Arrays
- Searching
- Files I/O
- Sorting
- Inheritance
- Recursion

Objectives of Lecture 11

Tracing Programs and the Debugger

- Learn how to trace the execution of a Java program.
- Understand what is happening during the execution of a program.
- Use program tracing:
  - to find errors in programs;
  - to understand what a program is supposed to do.
- Introduce the debugging facilities.

Outline of Lecture 11

- Example of a new program
- Notation for hand tracing
- Hand tracing Adventure
- The Code Warrior Debugger
- Tracing the example program again

public class Tunes {
   /* Creates a collection of CDs. Adds CDs to the collection and displays a summary of the collection. */
   public static void main(String args[]) {
      /* Program statements go here */
      CD_Collection music;
      music = new CD_Collection(5, 50.00f);
      music.addCDs(1, 10.99f);
      music.addCDs(3, 20.99f);
      music.displayCDs();
   }
}

class CD_Collection {
   /* Monitors the value of a collection of musical CDs. */
   private int numCDs;
   private float valueCDs;
   public CD_Collection (int initialNum, float initialVal) {
      /* Initializes the collection with the given number of CDs and the given value of the CD collection. */
      this.numCDs = initialNum;
      this.valueCDs = initialVal;
   }
   public void add_cds(int number, float value) {
      /* Adds CDs to the collection and adjusts the total value. */
      this.numCDs = this.numCDs + number;
      this.valueCDs = this.valueCDs + value;
   }
   public void displayCDs() {
      /* Displays the number of CDs in the collection and the total value of the collection. */
      System.out.println("================================");
      System.out.println("Total Number of CDs : " + this.numCDs);
      System.out.println("Total Value of Collection: " + this.valueCDs);
      System.out.println("Average cost per CD: $" + this.averageCost());
      System.out.println("================================");
   }
   private float averageCost() {
      /* Determines the average cost of a CD in the collection. */
      float average = this.valueCDs / this.numCDs;
      return average;
   }
}
Tracing

- Tracing is a technique that follows the execution of program in detail.
- Tracing can be used to understand how a Java program works.
- Tracing can also be used to find semantic errors in a program.
- A program can be hand traced by drawing diagrams.
- A program can also be traced using a tool called a debugger.

Notation for Hand Tracing

- Every method is represented by a rectangle.
- Every object is represented by an oval labeled by its class or its contents.
- Every reference is represented by a rectangle in the method that declares it.
- However, you can ignore public imported variables.
- Every reference has an arc connecting it to the object that it references.

Outline of Lecture 11

- Example of a new program
- Notation for hand tracing
  - Hand tracing Adventure
  - The Code Warrior Debugger
  - Tracing the example program again

Adventure Trace - call main

- Since this is an application, the interpreter invokes the static method called main.
- Since main is static, there is no - this.

Adventure Trace - main

- The parameter args is a reference
- The variable game is a reference

```
public static void main(String args[]) {
    Adventure game;
    game = new Adventure();
    game.play();
}
```

Adventure Trace - main - game

- When the new Adventure object is created we draw it and when the game reference is bound to the new object we connect it.

```
public static void main(String args[]) {
    Adventure game;
    game = new Adventure();
    game.play();
}
```
public static void main ...
    Adventure game;
    game = new Adventure();
    game.play();
}

private void play () {
    String name;
    Integer tokens;
    name = this.greeting();
    tokens = this.enterRoom(name);
    this.farewell(name, tokens);
}

private void greeting () {
    String playerName;
    System.out.println("Welcome to the Arithmetic ...
    The date is Tue February 1 ...
    What is your name?");
    System.out.print("What is …");
    playerName = Keyboard.in.readString();
    ... System.out.print("Welcome to the Arithmetic ...
    The date is Tue February 1 ...
    What is your name/");

Adventure Trace - greeting pause
• Output some more information and ask the keyboard to pause.
• Wait until the user presses the ENTER key.

```java
private String greeting() {
    String playerName;
    System.out.println("… air!");
    Keyboard.in.pause();
    return playerName;
}
```

Adventure Trace - greeting return
• Return the object bound to the variable `playerName` as the result of the message and discard the method.

```java
private String greeting() {
    String playerName;
    System.out.println("… air!");
    Keyboard.in.pause();
    return playerName;
}
```

Adventure Trace - play name
• Bind the variable `name` to the object that was returned from the greeting() message.

```java
private void play() {
    String name;
    Integer tokens;
    name = this.greeting();
    tokens = this.enterRoom(name);
    this.farewell(name, tokens);
}
```

Adventure Trace - call enterRoom
• When enterRoom() is sent to `this`, we draw an enterRoom() method with a new `this` reference, connect the methods and bind the new `this` to the receiver object.

```java
private Integer enterRoom(String theName) {
    Integer myTokens;
    System.out.print("How many…");
    System.out.print(theName);
    System.out.print("?");
    myTokens = Keyboard.in.readInteger();
    return myTokens;
}
```

Adventure Trace - enterRoom input
• Output some information, input an Integer from the keyboard and bind `myTokens` to it.

```java
private Integer enterRoom(String theName) {
    Integer myTokens;
    System.out.print("How many…");
    System.out.print(theName);
    System.out.print("?");
    myTokens = Keyboard.in.readInteger();
    return myTokens;
}
### Adventure Trace - enterRoom

**return**
- Return the object bound to the variable `myTokens` as the result of the message and discard the method.

### Adventure Trace - play tokens

**bind**
- Bind the variable `tokens` to the object that was returned from the `enterRoom()` message.

### Adventure Trace - call farewell

- When `farewell()` is sent to `this`, we draw a `farewell()` method with a new `this` reference, connect the methods and bind the new `this` to the receiver object.

### Adventure Trace - farewell output

- Output some information and ask the keyboard to pause.
- Wait until the user presses the ENTER key.

### Adventure Trace - farewell return

- This method does not return a result so just discard the method.
**Adventure Trace - play return**

- This method does not return a result so just discard the method.

```java
private void play() {
    String name;
    Integer tokens;
    name = this.greeting();
    tokens = this.enterRoom(name);
    this.farewell(name, tokens);
}
```

**Adventure Trace - main return**

- The static main method does not return a result so just discard the method.
- The program is now done.

```java
public static void main(...) {
    Adventure game;
    game = new Adventure();
    game.play();
}
```

---

**Outline of Lecture 11**

- Example of a new program
  - Notation for hand tracing
  - Hand tracing Adventure
  - The Code Warrior Debugger
  - Tracing the example program again

**Demonstration Debugger**

- Trace Adventure Version 2 in CodeWarrior using the debugger.
  - A demo of the debugger will be given in the lab.
  - The Debugger will allow you to execute your Java program statement by statement, and visualize your objects and variables during runtime.

---

**Outline of Lecture 11**

- Example of a new program
  - Notation for hand tracing
  - Hand tracing Adventure
  - The Code Warrior Debugger
  - Tracing the example program again

---

**Outline of Lecture 11**

- Example of a new program
  - Notation for hand tracing
  - Hand tracing Adventure
  - The Code Warrior Debugger
  - Tracing the example program again

---

**Outline of Lecture 11**

- Example of a new program
  - Notation for hand tracing
  - Hand tracing Adventure
  - The Code Warrior Debugger
  - Tracing the example program again
public class Tunes {

  /*
   * Creates a collection of CDs, adds CDs to the collection
   * and displays a summary of the collection value.
   */
  public static void main(String[] args) {
    CD_Collection music;
    music = new CD_Collection(5, 50.00f);
    music.addCDs(1, 10.99f);
    music.addCDs(3, 20.99f);
    music.displayCDs();
  }

  class CD_Collection {
    /*
     * Monitors the value of a collection of musical CDs.
     */
    private float averageCost() {
      /*
       * Determines the average cost of all CDs in the collection.
       */
      float average;
      average = this.valueCDs / this.numCDs;
      return average;
    }
  }
}

public class Tunes {

  /*
   * Creates a collection of CDs, adds CDs to the collection
   * and displays a summary of the collection value.
   */
  public static void main(String[] args) {
    CD_Collection music;
    music = new CD_Collection(5, 50.00f);
    music.addCDs(1, 10.99f);
    music.addCDs(3, 20.99f);
    music.displayCDs();
  }

  class CD_Collection {
    /*
     * Monitors the value of a collection of musical CDs.
     */
    private float averageCost() {
      /*
       * Determines the average cost of all CDs in the collection.
       */
      float average;
      average = this.valueCDs / this.numCDs;
      return average;
    }
  }
}

class CD_Collection {
  /*
   * Monitors the value of a collection of musical CDs.
   */
  ...