Selection Control Structures - if

Cmput 114 - Lecture 14

Department of Computing Science

University of Alberta

©Duane Szafron 1999

About This Lecture

- **Each of our programs consists of a sequence of statements.**
- **Even though we decompose our programs into methods, each method consists of a sequence of statements.**
- **We will learn how to write code that can conditionally execute a statement or not.**
- **We will also learn how to write code that can execute one statement or another statement.**

©Duane Szafron 1999

Outline

- **The boolean type**
- **The if statement**
- **The compound statement**
- **The if-then-else statement**
- **Adventure Version 5**

©Duane Szafron 1999

Sequence

- **Every method we have seen so far consists of a sequence of statements.**
- **The statements include: import statements, variable declaration statements, message expression statements, assignment statements, and return statements:**

```
String myString;
System.out.println("Java Rules!");
myString = keyboard.readString();
return name;
```

©Duane Szafron 1999

Selection

- **Sometimes it is useful to execute a statement or not, based on a condition.**
- **A statement that allows such conditional execution is called a selection statement.**
- **For example, the Adventure program can ask a user an arithmetic question:**
  - **If a user responds correctly, the program executes statements that reward the user.**
  - **If a user responds incorrectly, the program executes statements that penalize the user.**

©Duane Szafron 1999

The boolean Type

- **To support selection, many programming languages introduce a type called boolean, that has two members: *TRUE* and *FALSE*.**
- **Depending on the programming language, *TRUE* and *FALSE* may be objects or values.**
- **In Java, boolean is a primitive type.**
- **There are two literal booleans values: true and false.**

©Duane Szafron 1999

Creating booleans

- **Boolean values can be created by relational operators on int values:**
  greater or equal: 3 >= 4 --> false
  equal: index == 4 --> ?
- **boolean values can also be created by boolean operators on boolean values:**
  and: true && false --> false
  or: raining || snowing --> ?
- **There are many other operators that create booleans as well.**

©Duane Szafron 1999

Syntax for the Java if Statement

- **The syntax for an if statement in Java is:**
  <if statement> ::=
  if (<condition>) <statement>
- **A condition is any expression that evaluates to a boolean value.**
- **For example:**
  if (taxRate > 0.40f) amount = 2400;
- **The conventional format is:**
  if (this.chest != null)
  this.menu.add("Open the chest.");

©Duane Szafron 1999

Semantics for "if"

- **If the condition evaluates to true then the statement is executed.**
- **Otherwise the statement is skipped.**
- **For example, consider the statement:**
  ```
  if (this.chest != null)
    this.menu.add("Open the chest.");
  ```
- **If the current room's chest is not equal to `null` an item is added to the menu.**
- **Otherwise, the item is not added to the menu.**

The Java Compound Statement

- **Sometimes you want to execute more than one statement if the condition is satisfied.**
- **Java has a compound statement that can appear anywhere a statement can appear:**
  ```
  <compound statement> ::= \{ {<statement>} \}
  ```
- **Recall that {<xx>} is EBNF syntax that means zero or more occurrences of <xx>.**
- **The character \ is a meta-character that indicates the next symbol is a real symbol, not a meta-character!**

Example Compound Statement

- **For example,**
  ```
  if (action.equals("Open the chest.")) {
    this.chest.open(adventurer);
    this.chest = null;
  }
  ```
- **If the variable `action` is bound to a String that is equal to "Open the chest.", then an `open()` message is sent to the chest in the current room and then the chest variable is bound to `null`.**

Java if-then-else Statement

- **Java has another form of if statement called an if-else statement:**
  ```
  <if statement> ::= if (<condition>)
    <statement1> else <statement2>
  ```
  with semantics:
  - **If the condition is true then statement1 is executed and statement2 is skipped**
  - **If the condition is false then statement1 is skipped and statement2 is executed**

If-then-else Example

- Here is an example from the class `Chest`:

```
if (question.ask())
    this.correctAnswer(adventurer);
else
    this.wrongAnswer(question, adventurer);
```

- The message `ask()` returns a boolean that represents whether the user correctly answered the `question`.
- The chest then executes one of two methods to reward the adventurer or to penalize the adventurer.

Adventure Version 5

- We are going to add some functionality to the Arithmetic Adventure game .
- We will change the ask() method in class Question so it checks the user's answer against the correct answer and returns a boolean value true or false.
- We will change the open() method in the class Chest so that if the ask() messsage returns true then we will gain tokens and if returns false then we will remove tokens.

Adventure - Code Change Summary

- In the Chest class we will:
  - Modify the method open(Adventurer)
- In the Question class we will:
  - Modify the method ask()
- Leave the Adventure class unchanged except for changing the comment to Version 5.
- Leave the Adventurer class unchanged.

Running Adventure 5 (1)

Running Adventure 5 (2)


```
Java Console
Welcome to the Arithmetic Adventure game.
The date is Tue Apr 20 19:14:03 MDT 1999

What is your name?Fred
Well Fred, after a day of hiking you spot a silver cube.
The cube appears to be about 5 meters on each side.
You find a green door, open it and enter.
The door closes behind you with a soft whir and disappears.
There is a feel of mathematical magic in the air.
Press the ENTER key to continue.
There is a small carved chest in the center of the room.
It appears to be a treasure chest!
7 + 3 = 20

A small loudspeaker appears in the air.
You hear the sound of a deep gong and a pleasant voice says:
Sorry, the correct answer is 10.
You see 0 valuable tokens fly out of your pocket and fall to the floor.
A small vacuum cleaner appears, sweeps up your scattered tokens and disappears.
You have 0 tokens in your pocket.
Congratulations Fred you have left the game with 0 tokens.
```

©Duane Szafron 1999

NO CHANGES

Class - Chest 5.1

```java
import java.util.*;
public class Chest {
/*
   An instance of this class represents a treasure chest in
   the Adventure game. A Chest contains a number of tokens.
*/
/* Constructor */
   public Chest() {
   /*
       Initialize me so that I contain a random number of
       tokens.
   */
       this.tokens = Chest.generator.next(Chest.maxTokens);
   }
```

©Duane Szafron 1999

NO CHANGES

Class - Chest 5.2

```java
/* Instance Methods */

   public void display() {
   /*
       Output a description of myself.
   */

       System.out.println("There is a small carved chest in
the center of the room.");
       System.out.println("It appears to be a treasure
chest!");
   }
```

©Duane Szafron 1999

OLD

Class - Chest 4.3

```java
   public void open(Adventurer adventurer) {
   /*
       Ask the user an arithmetic question and if a correct
       answer is given, add tokens to the given Adventurer.
       If it is answered incorrectly, remove tokens.
   */
       Question question;

       question = new Question();
       question.ask();
       // We really want to do only one of the next two
       // lines, depending on the user's answer.
       this.correctAnswer(adventurer);
       this.wrongAnswer(question, adventurer);
   }
```

©Duane Szafron 1999

Class - Chest 5.3

```
public void open(Adventurer adventurer) {
/*
    Ask the user an arithmetic question and if a correct
    answer is given, add tokens to the given Adventurer.
    If it is answered incorrectly, remove tokens.
*/
    Question question;

    question = new Question();
    if (question.ask())
        this.correctAnswer(adventurer);
    else
        this.wrongAnswer(question, adventurer);
}
```

©Duane Szafron 1999

Class - Chest 5.4

```
/* Private Static Variables */

    private static final int maxTokens = 10;
    private static final RandomInt
        generator = new RandomInt(1);

/* Private Instance Variables */

    private int tokens;

/* Private Instance Methods */
```

©Duane Szafron 1999

Class - Chest 5.5

```
private void correctAnswer(Adventurer adventurer) {
/* Congratulate the adventurer and add some tokens.*/
    System.out.println();
    System.out.println("A small loudspeaker appears in the air.");
    System.out.println("You hear the sound of a harp and
a pleasant voice says congratulations.");
    System.out.print("The lid of the chest opens to reveal ");
    System.out.print(this.tokens);
    System.out.println(" valuable tokens.");
    System.out.println("They literally fly into your
pocket and the chest disappears.");
    adventurer.gainTokens(this.tokens);
    adventurer.reportTokens();
}
```

©Duane Szafron 1999

Class - Chest 5.6

```
private void wrongAnswer(Question question, Adventurer adventurer) {
/*
    Report the correct answer and remove some tokens
    from the given adventurer.
*/
    int   loss;

    System.out.println();
    System.out.println("A small loudspeaker appears in the air.");
    System.out.println("You hear the sound of a deep
gong and a pleasant voice says:");
    System.out.print("Sorry, the correct answer is ");
    System.out.print(question.answer());
    System.out.println(".");
```

©Duane Szafron 1999

Class - Chest 5.7

```
    loss = Math.min(this.tokens, adventurer.tokens());
    System.out.print("You see ");
    System.out.print(loss);
    System.out.println(" valuable tokens fly out of your
pocket and fall to the floor.");
    System.out.println("A small vacuum cleaner appears,
sweeps up your scattered tokens and disappears.");
    adventurer.loseTokens(loss);
    adventurer.reportTokens();
}
}
```

Class - Question 5.1

```
import java.util.*;
public class Question {
/*
  An instance of this class represents an arithmetic
  problem in the Arithmetic Adventure game.
*/

/* Constructor */
  public Question() {
  /*
      Initialize me so that I have two operands.
  */
      this.leftOperand = Question.generator.next(Question.maxOperand);
      this.rightOperand = Question.generator.next(Question.maxOperand);
  }
```

Class - Question 4.2

```
/* Instance Methods */
public void ask() {
/*
    Pose myself. Eventually I would like to indicate
    whether the user's response was correct or not.
*/
    Integer      answer;

    System.out.print(this.leftOperand);
    System.out.print(" + ");
    System.out.print(this.rightOperand);
    System.out.print(" = ");
    answer = Keyboard.in.readInteger();
}
```

Class - Question 5.2

```
/* Instance Methods */
public boolean ask() {
/*
    Pose myself. Return true if the user's response
    was correct and false otherwise.
*/
    Integer      answer;

    System.out.print(this.leftOperand);
    System.out.print(" + ");
    System.out.print(this.rightOperand);
    System.out.print(" = ");
    answer = Keyboard.in.readInteger();
    return answer.intValue() == this.answer();
}
```

Class - Question 5.3

```java
public int answer() {
/*
   Answer my correct answer.
*/
   return this.leftOperand + this.rightOperand;
}

/* Private Static Variables */
private static final int maxOperand = 9;
private static final RandomInt
   generator = new RandomInt(2);
/* Private Instance Variables */
private int leftOperand;
private int rightOperand;
```

No Changes to End of Lecture

- The rest of the Adventure program is included for completeness.
- There are no changes from the last version in the rest of these slides.

Program - Adventure 5.1

```java
import java.util.*;
public class Adventure {
/* Version 5
   This program is an arithmetic adventure game …
*/

/* Constructors */

   public Adventure () {
   /*
      Initialize an adventure by creating the appropriate
      objects.
   */
   }
```

Program - Adventure 5.2

```java
/* Main program */

   public static void main(String args[]) {
      Adventure   game;

      game = new Adventure();
      game.play();
   }
```

Program - Adventure 5.3

```java
/* Private Instance Methods */

  private void play() {
  /*
     Plays the Adventure game.
  */

     Adventurer  adventurer;

     adventurer = this.greeting();
     this.enterRoom(adventurer);
     this.farewell(adventurer);
  }
```

©Duane Szafron 1999

Program - Adventure 5.4

```java
  private Adventurer greeting() {
  /*
     Great the user and answer an Adventurer that
     represents the user.
  */
     String       playerName;

     System.out.println("Welcome to the Arithmetic Adventure game.");
     System.out.print("The date is ");
     System.out.println(new Date());
     System.out.println();
     System.out.print("What is your name?");
     playerName = Keyboard.in.readString();
```

©Duane Szafron 1999

Program - Adventure 5.5

```java
     System.out.print("Well ");
     System.out.print(playerName);
     System.out.println(", after a day of hiking you spot a silver cube.");
     System.out.println("The cube appears to be about 5 meters on each side.");
     System.out.println("You find a green door, open it and enter.");
     System.out.println("The door closes behind you with a soft whir and disappears.");
     System.out.println("There is a feel of mathematical magic in the air.");
     Keyboard.in.pause();
     return new Adventurer(playerName);
  }
```

©Duane Szafron 1999

Program - Adventure 5.6

```java
  private void enterRoom(Adventurer adventurer) {
  /*
     The given adventurer has entered the
     first room.
  */
     Chest        chest;

     chest = new Chest();
     chest.display();
     chest.open(adventurer);
  }
```

©Duane Szafron 1999

Program - Adventure 5.7

```java
private void farewell(Adventurer adventurer) {
/*
    Say farewell to the user and report the game result.
*/

    System.out.print("Congratulations ");
    System.out.print(adventurer.name());
    System.out.print(" you have left the game with ");
    System.out.print(adventurer.tokens());
    System.out.println(" tokens.");
}
```

Class - RandomInt 5.1

```java
import java.util.*;
public class RandomInt {
/*
  An instance of this class represents a generator that
  can generate a series of random positive ints.
*/

/* Contsructor */
  public RandomInt(int seed) {
  /*
      Initialize me so that I use the given seed.
  */
      this.generator = new Random(seed);
  }
```

Class - RandomInt 5.2

```java
/* Instance Methods */

  public int next(int max) {
  /*
      Answer a Random int between 1 and the given max.
  */

      return Math.round(max * this.generator.nextFloat() -
  0.5f) + 1;
  }

/* Private Instance Variables */
  private Random generator;
```

Class - Adventurer 5.1

```java
public class Adventurer {
/*
  An instance of this class represents a player of the
  Adventure game.
*/

/* Constructors */
  public Adventurer(String name) {
  /*
      Initialize me with the given name and zero tokens.
  */
      this.name = name;
      this.tokens = 0;
  }
```

Class - Adventurer 5.2

```
/* Instance Methods */

  public String name() {
  /*
     Answer a String representing my name.
  */
     return this.name;
  }

  public int tokens() {
  /*
     Answer my number of Tokens.
  */
     return this.tokens;
  }
```

©Duane Szafron 1999

Class - Adventurer 5.3

```
  public void gainTokens(int anInt) {
  /*
     Add the given number of tokens to my total.
  */
     this.tokens = this.tokens + anInt;
  }

  public void loseTokens(int anInt) {
  /*
     Remove the given number of tokens from my total.
  */
     this.tokens = this.tokens - anInt;
  }
```

©Duane Szafron 1999

Class - Adventurer 5.4

```
  public void reportTokens() {
  /*
     Output the number of tokens I have.
  */
     System.out.print("You have ");
     System.out.print(this.tokens);
     System.out.println(" tokens in your pocket.");
  }

/* Private Instance Variables */

  private String name;
  private int tokens;
```

©Duane Szafron 1999

Selection Control Structures - switch

Cmput 114 - Lecture 15

Department of Computing Science

University of Alberta

©Duane Szafron 1999

About This Lecture

- **In this lecture we will study another selection control structure called the switch statement.**
- **A switch statement conditionally executes one of many statements depending on a selection value.**

Outline

- **The switch statement**
- **Adventure Version 6**

Java switch Statement Syntax

```
switch (<expression>) {
    case <value11>: case <value12>: …
        <statement1 block>
    case <value21>: case <value22>: …
        <statement2 block>
    …
    default:
        <defaultStatement block>
}
```

- A **condition** is any expression that evaluates to a boolean value.
- For example:
    if (taxRate > 0.40f) amount = 2400;
- The conventional format is:
    if (this.chest() != null)
        menu.add("Open the chest.");

Switch Statement Semantics 1

- **If the expression evaluates to value11 or value12 then the statements in the first block are executed.**
- **If the expression evaluates to value21 or value22 then the statements in the second block are executed.**
- **If the value of the expression does not equal any value in any of the case clauses then the default statement block is executed.**

Switch Statement Semantics 2

- **If a statement block is executed and it contains the statement:**

  break;

  then the rest of the statement blocks are skipped.
- **If a statement block is executed and it does NOT contain a break statement, then all of the following statement blocks are executed, regardless of the case values.**

Java switch Example

```
switch (index) {
    case 1:
        string = this.entry1;
        break;
    case 2:
        string = this.entry2;
        break;
    …
    case 5:
        string = this.entry5;
        break;
    default:
        string = this.entry1;
        break;
}
```

Another Java switch Example

```
switch (index) {
    case 1: return this.entry1;
    case 2: return this.entry2;
    …
    case 5: return this.entry5;
    default: return this.entry1;
}
```

Adventure 6

- **We are going to add some functionality to the Arithmetic Adventure game .**
- **We are going to create a room for the Adventurer and allow the user to either open the chest in the room or quit.**
- **To do this we will create two new classes, Room and TextMenu.**
- **Later we will add more rooms, add doors to the room and allow the user to open a door and enter another room.**

Adventure - Code Change Summary

- **In the Adventure class we will:**
  - **Modify the constructor Adventure() by adding code.**
  - **Add an an instance variable that is bound to the first Room object that the user enters.**
  - **Modify the play() method.**
- **Create the new class Room.**
- **Create the new class TextMenu.**
- **Leave the classes: Question, Chest, RandomInt and Adventurer unchanged.**

©Duane Szafron 1999

Adventure 6 (1)
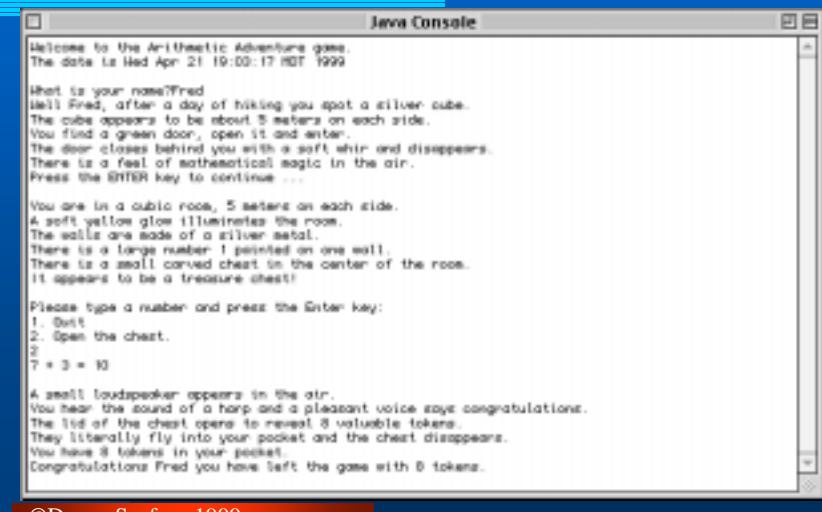
©Duane Szafron 1999

Adventure 6 (2)

©Duane Szafron 1999

Program - Adventure 5.1

```java
import java.util.*;
public class Adventure {
/* Version 5
   This program is an arithmetic adventure game …
*/

/* Constructors */

   public Adventure () {
   /*
      Initialize an adventure by creating the appropriate
      objects.
   */
   }
```

©Duane Szafron 1999

Program - Adventure 6.1

```java
import java.util.*;
public class Adventure {
/* Version 6
   This program is an arithmetic adventure game …
*/

/* Constructors */

   public Adventure () {
   /*
      Initialize an adventure by creating the appropriate
      objects.
   */
      this.firstRoom = new Room(1);
}
```

©Duane Szafron 1999

Program - Adventure 5.2

```java
/* Main program */

   public static void main(String args[]) {
      Adventure   game;

      game = new Adventure();
      game.play();
}
```

©Duane Szafron 1999

Program - Adventure 6.2

```java
/* Main program */

   public static void main(String args[]) {
      Adventure   game;

      game = new Adventure();
      game.play();
   }

/* Private Instance Variables */

private Room firstRoom;
```

©Duane Szafron 1999

Program - Adventure 5.3

```java
/* Private Instance Methods */

   private void play() {
   /*
      Plays the Adventure game.
   */

      Adventurer   adventurer;

      adventurer = this.greeting();
      this.enterRoom(adventurer);
      this.farewell(adventurer);
}
```

©Duane Szafron 1999

Program - Adventure 6.3

```
/* Private Instance Methods */

  private void play() {
  /*
      Plays the Adventure game.
  */

      Adventurer   adventurer;
      Room         room;

      adventurer = this.greeting();
      room = firstRoom.enter(adventurer);
      this.farewell(adventurer);
  }
```

Program - Adventure 6.4

```
  private Adventurer greeting() {
  /*
      Great the user and answer an Adventurer that
      represents the user.
  */
      String       playerName;

      System.out.println("Welcome to the Arithmetic Adventure game.");
      System.out.print("The date is ");
      System.out.println(new Date());
      System.out.println();
      System.out.print("What is your name?");
      playerName = Keyboard.in.readString();
```

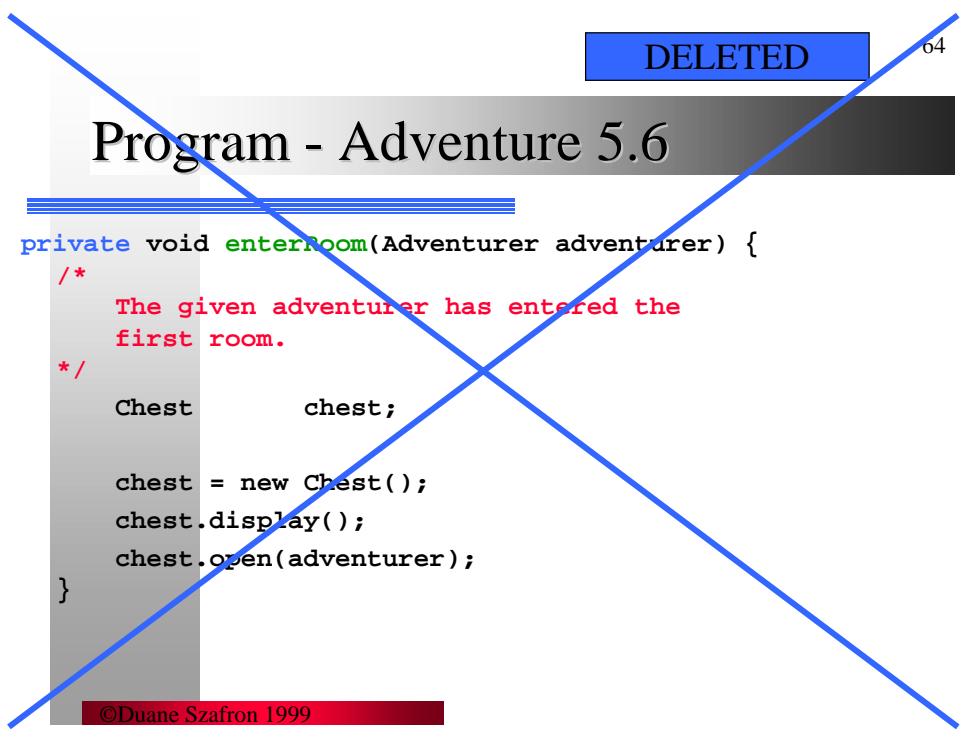## Program - Adventure 6.5

```
      System.out.print("Well ");
      System.out.print(playerName);
      System.out.println(", after a day of hiking you spot a silver cube.");
      System.out.println("The cube appears to be about 5 meters on each side.");
      System.out.println("You find a green door, open it and enter.");
      System.out.println("The door closes behind you with a soft whir and disappears.");
      System.out.println("There is a feel of mathematical magic in the air.");
      Keyboard.in.pause();
      return new Adventurer(playerName);
  }
```

Program - Adventure 5.6

```
  private void enterRoom(Adventurer adventurer) {
  /*
      The given adventurer has entered the
      first room.
  */
      Chest        chest;

      chest = new Chest();
      chest.display();
      chest.open(adventurer);
  }
```

Program - Adventure 6.7

```java
private void farewell(Adventurer adventurer) {
/*
    Say farewell to the user and report the game result.
*/

    System.out.print("Congratulations ");
    System.out.print(adventurer.name());
    System.out.print(" you have left the game with ");
    System.out.print(adventurer.tokens());
    System.out.println(" tokens.");
}
```

©Duane Szafron 1999

Class - Room 6.1

```java
import java.util.*;
public class Room {
/*
  A room contains a treasure chest and some doors to
  adjoining rooms.
*/
/* Constructor */
  public Room(int anInt) {
  /*
      Initialize me so that I have the given room number,
      contain a treasure chest, and no doors.
  */
      this.number = anInt;
      this.chest = new Chest();
  }
```

©Duane Szafron 1999

Class - Room 6.2

```java
/* Instance Methods */
  public Room enter(Adventurer adventurer) {
  /*
      Describe myself, display a list of options, and
      perform the selected option. If the user selected
      quit then return null. If the user selected to go
      to another Room then return that Room. Otherwise
      return this Room.
  */
      TextMenu    menu;
      String      action;
      this.display();
      menu = this.buildMenu();
      action = menu.launch();
      return this.performAction(action, adventurer);
} 
```
©Duane Szafron 1999

Class - Room 6.3

```java
/* Private Instance Variables */
  private Chest    chest;
  private int      number;

/* Private Instance Methods */

  private void display() {
  /*
      Output a description of myself.
  */
      this.displayBasic();
      this.displayDoors();
      if (this.chest != null)
          this.chest.display();
  }
```
©Duane Szafron 1999

Class - Room 6.4

```java
private void displayBasic() {
/*
    Output a basic description of myself that is
    independent of my doors and contents.
*/
    System.out.println();
    System.out.println("You are in a cubic room, 5 meters on each side.");
    System.out.println("A soft yellow glow illuminates the room.");
    System.out.println("The walls are made of a silver metal.");
    System.out.print("There is a large number ");
    System.out.print(this.number);
    System.out.println(" painted on one wall.");
}
```

Class - Room 6.5

```java
private void displayDoors() {
/*
    Output a description of all of my doors.
*/
}
```

Class - Room 6.6

```java
private TextMenu buildMenu() {
/*
    Create and answer a TextMenu containing the user's
    valid actions.
*/
    TextMenu    menu;

    menu = new TextMenu();
    menu.add("Quit");
    if (this.chest != null)
        menu.add("Open the chest.");
    // Add door choices here
    return menu;
}
```

Class - Room 6.7

```java
private Room performAction(String action, Adventurer adventurer) {
/*  Perform the action described by the given String for
    the given Adventurer. Return the room the user
    selected, null if the user selected quit and this
    room if the user selected to open the chest.
*/
    if (action.equals("Open the chest.")) {
        this.chest.open(adventurer);
        this.chest = null;
        return this;
    }
    if (action.equals("Quit"))
        return null;
    return null;
}
}
```

Class - TextMenu 6.1

```java
import java.io.*;
import java.util.*;
public class TextMenu {
/*
   An instance of this class displays a list of strings for
   the user and allows the user to pick one. For now, up to
   five entries are supported.
*/
/* Contructor */

   public TextMenu() {
   /*
      Initialize me with no entries.
   */
   }
```

©Duane Szafron 1999

Class - TextMenu 6.2

```java
/* Instance Methods */
   public void add(String entry) {
   /*
      Add the given String to me as my next choice.
   */
      if (entry1 == null) {
         this.entry1 = entry;
         return;
      }
      if (entry2 == null) {
         this.entry2 = entry;
         return;
      }
   //more of the same for entries 3, 4 and 5.
   }
```

©Duane Szafron 1999

Class - TextMenu 6.3

```java
   public String launch() {
   /*
      Display myself and answer the String entry selected
      by the user.
   */
      Integer    choice;
      int        index;

      this.display();
      choice = Keyboard.in.readInteger();
      if (choice == null)
         return this.entry1;
      index = choice.intValue();
```

©Duane Szafron 1999

Class - TextMenu 6.4

```java
      switch (index) {
         case 1: return this.entry1;
         case 2: return this.entry2;
         case 3: return this.entry3;
         case 4: return this.entry4;
         case 5: return this.entry5;
         default: return this.entry1;
      }
   }
```

©Duane Szafron 1999

Class - TextMenu 6.5

```
/* Private Instance Variables */

  private String entry1;
  private String entry2;
  private String entry3;
  private String entry4;
  private String entry5;

/* Private Instance Methods */
```

Class - TextMenu 6.6

```
  private void display() {
  /*
    Display myself on the screen.
  */
    String      entry;
    int         index;
    System.out.println();
    System.out.println("Please type a number and press the Enter key:");
    if (this.entry1 != null) {
        System.out.print("1. ");
        System.out.println(this.entry1);
    }
    // same code for entry2, entry3, entry4 and entry5
  }
}
```

No Changes to End of Lecture

- The rest of the Adventure program is included for completeness.
- There are no changes from the last version in the rest of these slides.

NO CHANGES

Class - Chest 6.1

```
import java.util.*;
public class Chest {
/*
  An instance of this class represents a treasure chest in
  the Adventure game. A Chest contains a number of tokens.
*/
/* Constructor */
  public Chest() {
  /*
    Initialize me so that I contain a random number of
    tokens.
  */
    this.tokens = Chest.generator.next(Chest.maxTokens);
  }
```

Class - Chest 6.2

```java
/* Instance Methods */

  public void display() {
  /*
    Output a description of myself.
  */

    System.out.println("There is a small carved chest in
the center of the room.");
    System.out.println("It appears to be a treasure
chest!");
  }
```

Class - Chest 6.3

```java
public void open(Adventurer adventurer) {
/*
    Ask the user an arithmetic question and if a correct
    answer is given, add tokens to the given Adventurer.
    If it is answered incorrectly, remove tokens.
*/
    Question question;

    question = new Question();
    if (question.ask())
        this.correctAnswer(adventurer);
    else
        this.wrongAnswer(question, adventurer);
}
```

Class - Chest 6.4

```java
/* Private Static Variables */

  private static final int maxTokens = 10;
  private static final RandomInt
      generator = new RandomInt(1);

/* Private Instance Variables */

  private int tokens;

/* Private Instance Methods */
```

Class - Chest 6.5

```java
private void correctAnswer(Adventurer adventurer) {
/* Congratulate the adventurer and add some tokens.*/
    System.out.println();
    System.out.println("A small loudspeaker appears in the air.");
    System.out.println("You hear the sound of a harp and
a pleasant voice says congratulations.");
    System.out.print("The lid of the chest opens to reveal ");
    System.out.print(this.tokens);
    System.out.println(" valuable tokens.");
    System.out.println("They literally fly into your
pocket and the chest disappears.");
    adventurer.gainTokens(this.tokens);
    adventurer.reportTokens();
}
```

Class - Chest 6.6

```java
private void wrongAnswer(Question question, Adventurer adventurer) {
/*
    Report the correct answer and remove some tokens
    from the given adventurer.
*/
    int   loss;

    System.out.println();
    System.out.println("A small loudspeaker appears in the air.");
    System.out.println("You hear the sound of a deep
gong and a pleasant voice says:");
    System.out.print("Sorry, the correct answer is ");
    System.out.print(question.answer());
    System.out.println(".");
```

Class - Chest 6.7

```java
    loss = Math.min(this.tokens, adventurer.tokens());
    System.out.print("You see ");
    System.out.print(loss);
    System.out.println(" valuable tokens fly out of your
pocket and fall to the floor.");
    System.out.println("A small vacuum cleaner appears,
sweeps up your scattered tokens and disappears.");
    adventurer.loseTokens(loss);
    adventurer.reportTokens();
}
}
```

Class - Question 6.1

```java
import java.util.*;
public class Question {
/*
  An instance of this class represents an arithmetic
  problem in the Arithmetic Adventure game.
*/

/* Constructor */
public Question() {
/*
    Initialize me so that I have two operands.
*/
    this.leftOperand = Question.generator.next(Question.maxOperand);
    this.rightOperand = Question.generator.next(Question.maxOperand);
}
```

Class - Question 6.2

```java
/* Instance Methods */
public boolean ask() {
/*
    Pose myself. Return true if the user's response
    was correct and false otherwise.
*/
    Integer     answer;

    System.out.print(this.leftOperand);
    System.out.print(" + ");
    System.out.print(this.rightOperand);
    System.out.print(" = ");
    answer = Keyboard.in.readInteger();
    return answer.intValue() == this.answer();
}
```

Class - Question 6.3

```java
public int answer() {
/*
    Answer my correct answer.
*/
    return this.leftOperand + this.rightOperand;
}

/* Private Static Variables */
private static final int maxOperand = 9;
private static final RandomInt
    generator = new RandomInt(2);
/* Private Instance Variables */
private int leftOperand;
private int rightOperand;
```

©Duane Szafron 1999

Class - RandomInt 6.1

```java
import java.util.*;
public class RandomInt {
/*
    An instance of this class represents a generator that
    can generate a series of random positive ints.
*/

/* Contsructor */
public RandomInt(int seed) {
/*
    Initialize me so that I use the given seed.
*/
    this.generator = new Random(seed);
}
```

©Duane Szafron 1999

Class - RandomInt 6.2

```java
/* Instance Methods */

public int next(int max) {
/*
    Answer a Random int between 1 and the given max.
*/

    return Math.round(max * this.generator.nextFloat() -
0.5f) + 1;
}

/* Private Instance Variables */
private Random generator;
```

©Duane Szafron 1999

Class - Adventurer 6.1

```java
public class Adventurer {
/*
    An instance of this class represents a player of the
    Adventure game.
*/

/* Constructors */
public Adventurer(String name) {
/*
    Initialize me with the given name and zero tokens.
*/
    this.name = name;
    this.tokens = 0;
}
```

©Duane Szafron 1999

Class - Adventurer 6.2

```java
/* Instance Methods */

  public String name() {
  /*
     Answer a String representing my name.
  */
     return this.name;
  }

  public int tokens() {
  /*
     Answer my number of Tokens.
  */
     return this.tokens;
  }
```

©Duane Szafron 1999

Class - Adventurer 6.3

```java
  public void gainTokens(int anInt) {
  /*
     Add the given number of tokens to my total.
  */
     this.tokens = this.tokens + anInt;
  }

  public void loseTokens(int anInt) {
  /*
     Remove the given number of tokens from my total.
  */
     this.tokens = this.tokens - anInt;
  }
```

©Duane Szafron 1999

Class - Adventurer 6.4

```java
  public void reportTokens() {
  /*
     Output the number of tokens I have.
  */
     System.out.print("You have ");
     System.out.print(this.tokens);
     System.out.println(" tokens in your pocket.");
  }

/* Private Instance Variables */

  private String name;
  private int tokens;
```

©Duane Szafron 1999