So far our programs execute each statement exactly once or zero times, if a selection control structure is used. In this lecture we will learn how to write programs in which statements can be executed many times using an indefinite repetition control structure called the while statement.

About This Lecture

- The while statement
- Input validation
- Adventure Version 7

Sometimes it is useful to execute a block of several statements more than once. A control structure that supports this is called a repetition control structure.
Java Syntax: while Statement

- The syntax for a while statement in Java is:
  
  `<while statement> ::= while (<condition>) <statement>

- For example:
  
  ```java
  i = 0;
  while (i <= 4) {
    System.out.println(i);
    i = i + 1;
  }
  System.out.println("That’s all folks");
  ```

Semantics - while

- If the condition evaluates to true then the statement is executed.
- The condition is then evaluated again.
- If the condition is still true then the statement is executed again.
- This continues until the condition evaluates to false.
- Control then goes to the statement after the while statement.

While Semantics - Example

```java
i = 0;
while (i <= 4) {
  System.out.println(i);
  i = i + 1;
}
System.out.println("That’s all folks");
```

Input Validation - Question

- What happens in the Adventure game if a Question asks the user for an answer and the user types a String that does not represent an integer?
Question Input Validation Error

/* Instance Methods */
public boolean ask () {
    /* Pose myself and answer true or false depending on whether the user answers correctly or not. */
    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);
}

answer may be bound to null

Input Validation - TextMenu

- What happens in the Adventure game if a TextMenu queries the user for an answer and the user makes an invalid choice?

TextMenu Input Validation Error
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;
}

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);
}

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();
    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;
    }
}
Validate User Input Using while

- A while statement can be used to check user input and to re-query until valid input is entered.
- The general format of this approach is:

  ```java
  answer = null;
  while (answer == null) {
     //display prompt
     answer = //valid answer or null
  }
  ```

while in Class Question

```java
answer = null;
while (answer == null) {
    this.display();
    answer = keyboard.readInteger();
}
return (answer.intValue() == this.answer());
```
- Read strings typed by the user until the user enters a string that can be converted to an Integer.
- Exit the while statement and then return whether it is the correct answer or not.

while in class TextMenu

```java
index = 0;
while (((index < 1) | (index > this.size)) {
    this.display();
    choice = Keyboard.in.readInteger();
    if (choice == null)
        index = 0;
    else
        index = choice.intValue();
}
```
- Read strings typed by the user until the user enters a string that can be converted to an Integer between 1 and some size.

Adventure 7

- Modify the Arithmetic Adventure game to do input validation.
Adventure - Changes Summary 1

- In the TextMenu class we will:
  - Add an instance variable called size which indicates how many legal entries I have.
  - Modify the constructor TextMenu().
  - Modify the instance method add().
  - Replace instance method launch().
  - Add instance method getUserSelection().

Adventure - Changes Summary 2

- In the Question class we will:
  - Replace the ask() method.
  - Add a display() method.
- Leave the classes: Adventure, Adventurer, RandomInt, Chest and Room unchanged.

Running Adventure 7

```java
import java.util.*;
public class Question {
    /*
    An instance of this class represents an arithmetic problem in the Arithmetic Adventure game.
    */
    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 7.1
/** 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();
}

/* 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;

/* Instance Methods */
public boolean ask () {
    /* Pose myself. Return true if the user’s response was correct and false otherwise. */
    Integer answer;
    answer = null;
    while (answer == null) {
        this.display();answer = Keyboard.in.readInteger();
    }
    return answer.intValue() == this.answer();
}

/* Private Instance Methods */
public void display () {
    /* Display myself. */
    System.out.print(this.leftOperand);
    System.out.print(" + ");System.out.print(this.rightOperand);
    System.out.print(" = ");
}

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;
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.
     */
    constructor *
    public TextMenu() {
        this.size = 0;
        /*
         * Initialize me with no entries.
         */
    }
}

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

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;
}

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

    index = this.getUserSelection();

    switch (index) {
        case 1: action = this.entry1; break;
        case 2: action = this.entry2; break;
        case 3: action = this.entry3; break;
        case 4: action = this.entry4; break;
        case 5: action = this.entry5; break;
        default: action = "";
    }

    return action;
}
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 7.5

/* Private Instance Variables */
private String entry1;
private String entry2;
private String entry3;
private String entry4;
private String entry5;
private int size;

/* Private Instance Methods */

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
    }

Class - TextMenu 7.6

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

Class - TextMenu 7.7

private int getUserSelection () {
    /*
    Query the user for an action and answer the index of the choice. If the user does not answer with a valid action, query again.
    */
    Integer choice;
    int index;
    index = 0;
    index = 0;
while ((index < 1) || (index > this.size)) {
    this.display();
    choice = Keyboard.in.readInteger();
    if (choice == null)
        index = 0;
    else
        index = choice.intValue();
}
return index;

The rest of the classes are omitted to save space.
See Lecture 15 for a listing.