Learn more about our research, discover data science, and find other great resources at:

http://www.dataminingapps.com

Chapter 5
Controlling the Flow of Your Program

Overview

- Comparisons using Operators and Methods
- Understanding Language Control
- Reviewing Keywords for Control

Comparisons using Operators and Methods

- Primitive data types
  - Relational + logical operators (see Chapter 2)
- Composite data types
  - Comparison methods must be defined the class
  - E.g. Strings

```
String myString = "I'm a string.";
String anotherString = "I'm a string, too.";
String oneMoreString = "I'm a string.";
myString.equals(oneMoreString); //this will evaluate as TRUE
myString.equals(anotherString); //this will evaluate as FALSE
```

Comparisons using Operators and Methods

- Note: == versus `equals()`
  - ==: checks whether two objects are the same (i.e. same position in memory)
  - `equals()` : used to e.g. check if two Strings have the same content

Understanding Language Control

- Creating `if-then` Statements
- Creating `for` loops
- Creating `while` loops
- Comparing `for` and `while` loops
- Creating switches

Creating `if-then` Statements

```
if (accountBalance > 100) {
System.out.println("Safe balance.");
}

if (accountBalance > 100) {
   System.out.println("Safe balance.");
} else {
   System.out.println("Warning: Low balance.");
}

if (accountBalance > 100) {
   System.out.println("Safe balance.");
} else if (accountBalance < 0){
   System.out.println("ALERT: Negative balance!");
} else {
   System.out.println("Warning: Low balance.");
}
```

Creating `if-then` Statements

```
if (accountBalance > 0) {
   System.out.println("Safe balance.");
   if (accountDays > 90) {
       System.out.println(savingsAccountOffer);
       if (creditAccounts > 1) {
          balanceTransferPossible = true;
       } else {
          sendCreditApplication();
       }
   }
 } else {
    System.out.println("ALERT: Negative balance!");
 }
```

Creating `if-then` Statements

```
if (accountBalance > 0 && accountDays <= 90) {
   System.out.println("Safe balance.");
} else if (accountBalance > 0 && accountDays > 90 && creditAccounts > 1) {
   System.out.println("Safe balance.");
   System.out.println(savingsAccountOffer);
   balanceTransferPossible = true;
} else if (accountBalance > 0 && accountDays > 90) {
   System.out.println("Safe balance.");
   System.out.println(savingsAccountOffer);
   sendCreditApplication();
}
} else {
   System.out.println("ALERT: Negative balance!");
}
```

Creating `for` loops

- `for loop` is a structure that cycles through a section of code as long as some condition is met
- Syntax:

```
for (/*Initialization*/; /*Termination*/; /*Increment*/){
/*execute these statements*/
}
```

- Key components:
  - Initialization: declares index variable and its starting value (e.g. `int i=0`)
  - Termination: specifies stopping condition or maximum value for index
  - Increment: indicates how index value should change after each iteration (e.g. `i++`)

Creating for loops

```
int[] sales2014 =
{500,720,515,377,400,435,510,1010,894,765,992,1125};
int[] staff2014 = {7,5,5,5,5,6,6,7,7,8,9,9};
int[] salesPerStaff = new int[12];
int totalSales2014 = 0;

for (int i=0; i<sales2014.length; i++){
    salesPerStaff[i] = sales2014[i]/staff2014[i];
    totalSales2014 = totalSales2014 + sales2014[i];
}
```

Creating for loops

```
public class ForLoop {
public static void main(String[] args){
  for (int i = 5; i > 0; i++){
     System.out.println(“Greetings from loop");
  };
}}
```

```
public class ForLoop {
public static void main(String[] args){
   for (int i = 1; i <= 5; i++){
      int doubled = i * 2;
      System.out.println(i + " times two equals " + doubled);
   }
   System.out.println("End of program");
}}
```

Creating for loops

```
public class ForLoop {
public static void main(String[] args){
 for (int i = 5; i > 0; i++){
   System.out.println(“Greetings from loop");
 };
}}
```

Output:

```
Greetings from loop
Greetings from loop
Greetings from loop
Greetings from loop
…
```

Infinite loop!

```
public class ForLoop {
public static void main(String[] args){
for (int i = 1; i <= 5; i++){
int doubled = i * 2;
System.out.println(i + " times two equals " + doubled);
}
System.out.println("End of program");
}}
```

Output:

1 times two equals 2
2 times two equals 4
3 times two equals 6
4 times two equals 8
5 times two equals 10
End of program

Creating for loops

- Enhanced for loop
  - Created for arrays and other iterable objects
  - Iterator automatically iterates through elements
- Examples:

```
for (int i: sales2014){
   salesPerStaff[i] = sales2014[i]/staff2014[i];
   totalSales2014 = totalSales2014 + sales2014[i];
}

String[] nameList = {"Adam Brown","Betsy Dudley","Carl Frank"};
for (String name: nameList){
   System.out.println(name);
}
```

Creating for loops

Employee	Monday	Tuesday	Wednesday	Thursday	Friday
Bart	3	2	8	2	3
Seppe	4	4	4	4	4
Aimée	5	5	0	5	5

```
int[][] hoursWorked = {{3,2,8,2,3},{4,4,4,4,4,4},{5,5,0,5,5}};
String[] employees = {“Bart", “Seppe", “Aimée"};
double wage = 8.30;
for (int i = 0; i < hoursWorked.length; i++){ //outer for loop
    System.out.print(employees[i] + " worked ");
    int weeklyHours = 0;
    for (int j = 0; j < hoursWorked[0].length; j++){ //inner for loop
        weeklyHours += hoursWorked[i][j];
     } //close inner for loop
    System.out.println(weeklyHours + " hours at " + wage + " per hour.");
    double weeklyPay = weeklyHours * wage;
    System.out.println("Weekly Pay: " + weeklyPay);
} //close outer for loop
```

Creating for loops

Employee	Monday	Tuesday	Wednesday	Thursday	Friday
Bart	3	2	8	2	3
Seppe	4	4	4	4	4
Aimée	5	5	0	5	5

```
int[][] hoursWorked = {{3,2,8,2,3},{4,4,4,4,4,4},{5,5,0,5,5}};
String[] employees = {“Bart", “Seppe", “Aimée"};
double wage = 8.30;
for (int i = 0; i < hoursWorked.length; i++){ //outer for loop
    System.out.print(employees[i] + " worked ");
    int weeklyHours = 0;
    for (int j = 0; j < hoursWorked[0].length; j++){ //inner for loop
        weeklyHours += hoursWorked[i][j];
     } //close inner for loop
    System.out.println(weeklyHours + " hours at " + wage + " per hour.");
    double weeklyPay = weeklyHours * wage;
    System.out.println("Weekly Pay: " + weeklyPay);
} //close outer for loop
```

Output:

```
Bart worked 18 hours at 8.3 per hour.
Weekly Pay: 149.4
Seppe worked 20 hours at 8.3 per hour.
Weekly Pay: 166.0
Aimée worked 20 hours at 8.3 per hour.
Weekly Pay: 166.0
```

Creating `while` loops

- Alternative loop structure based on meeting a certain condition
- Syntax:

```
while (/*conditional expression*/) {
/*execute these statements*/
}
```

- Example

```
int i = 5;
while (i > 0){
   System.out.println(i);
   i = i - 1;
}
```

Output:

```
5
4
3
2
1
```

Creating `while` loops

```
public class WhileLoop {
public static void main(String[] args) {
int i = 5;
while (i > 0){
   System.out.println(“Greetings from the Loop”);
}}
```

Output:

```
Greetings from loop
Greetings from loop
Greetings from loop
Greetings from loop
…
```

Infinite loop!

```
public class WhileLoop {
public static void main(String[] args) {
int i = 1;
while (i <= 5){
   int doubled = i * 2;
   System.out.println(i + " times two equals " + doubled);
   i++;
}
System.out.println("End of program");
}}
```

Output:

1 times two equals 2
2 times two equals 4
3 times two equals 6
4 times two equals 8
5 times two equals 10
End of program

Creating `while` loops

- `do while` loop
- Will first execute statements and then check condition (↔ `while` loop)
- Syntax:

```
do {
   /*execute these statements*/
} while (/*conditional expression*/);
```

- Example:

```
int i = 5;
do {
   System.out.println(i);
   i = i - 1;
} while (i > 0);
```

Output:

5
4
3
2
1

Creating `while` loops

```
public class DoWhileLoop {
public static void main(String[] args) {
int i = 1;
do {
   int doubled = i * 2;
   System.out.println(i + " times two equals " + doubled);
   i++;
} while (i <= 5);
System.out.println("End of program");
}
}
```

Output:

1 times two equals 2
2 times two equals 4
3 times two equals 6
4 times two equals 8
5 times two equals 10
End of program

Comparing `for` and `while` loops

- Guidelines
  - If you are iterating over a collection, consider a `for` loop first.
  - If you know the number of loops in advance, consider a `for` loop first.
  - If you don’t know the number of iterations, but the number depends on a certain condition, consider a `while` loop first.

Creating Switches

- Similar to `if-then` statement
- Useful in case of several `else` statements
- Syntax:

```
switch (/*variable*/ {
case 1: /*execute these statements*/; break;
case 2: /*execute these statements*/; break;
default: /*execute these statements*/;
}
```

Creating Switches

```
char initial;
String myName = "Bob";
switch (myName) {
case "Ann": initial = 'A'; break;
case "Bob": initial = 'B'; break;
case "Claire": initial = 'C'; break;
default: initial = '?'; break;
}
```

Creating Switches

```
int month = 4; //April
int lastDay;
boolean leapYear = false;
switch (month) {
case 1: lastDay = 31; break;
case 2: if (leapYear == true) {
lastDay = 29;
} else {
lastDay = 28;
} break;
case 3: lastDay = 31; break;
case 4: lastDay = 30; break;
case 5: lastDay = 31; break;
case 6: lastDay = 30; break;
…
```

```
case 7: lastDay = 31; break;
case 8: lastDay = 31; break;
case 9: lastDay = 30; break;
case 10: lastDay = 31; break;
case 11: lastDay = 30; break;
case 12: lastDay = 31; break;
default: lastDay = 0;
```

Creating Switches

```
int month = 4; // April
int lastDay;
boolean leapYear = false;
switch (month) {
case 1:
case 3:
case 5:
case 7:
case 8:
case 10:
case 12:
lastDay = 31; break;
…
```

```
case 2:
if (leapYear == true) {
lastDay = 29;
} else {
lastDay = 28;
} break;
case 4:
case 6:
case 9:
case 11:
lastDay = 30; break;
default: lastDay = 0;
}
```

Creating switches

```
public class SwitchClass {
   public static void main(String[] args) {
     String loanType = "Commercial";
     double interestRate;
     switch (loanType) {
     case "Residential":
       interestRate = 0.055;
       break;
     case "Commercial":
       interestRate = 0.062;
       break;
     case "Investment":
       interestRate = 0.059;
       break;
     default:interestRate = 0;
}
System.out.println(loanType + " loans have an annual interest rate of "
+ interestRate * 100 + "%.");
}}
```

Output:

Commercial loans have an annual interest rate of 6.2%.

Comparing switches and `if-then` statements

- Guidelines
  - If you have a single variable that can take multiple values, a switch might be suitable.
  - If you have multiple variables or conditions to consider, you will probably need an `if-then` statement.
  - If the variable you are considering can have a finite number of values, consider using a switch.
  - If the variable can take any value within a continuous range of numbers, consider an `if-then` statement.

Reviewing Keywords for Control

- `return`
  - Completes the execution of a method by returning a value
- `break`
  - Interrupts the execution of the current block of code (e.g. loop) but method execution continues
- `continue`
  - stops the current interation of the loop and continues with the next iteration

Reviewing Keywords for Control

```
//array of employee ID numbers, stored as Strings
static String[] employees;

//method to search for a specified employee ID
static boolean findEmployee(String employeeID){
   for (String emp : employees){
      if (emp.equals(employeeID)){
        return true;
      }
   }
  return false;
}
```

Reviewing Keywords for Control

```
import java.util.ArrayList;

static ArrayList<String> employeeList = new ArrayList< >();

//a method to add new Employees to the Employee array
static void addNewEmployee(String employeeID){
   if (employeeList.contains(employeeID)){
      return; //employee already exists
   }
 employeeList.add(employeeID);
}
```

Reviewing Keywords for Control

```
//array of employee ID numbers, stored as Strings
static String[] employees;

//method to search for a specified employee ID
static void findEmployee(String employeeID){
  String myString = employeeID + " was not found.";
  for (String emp : employees){
     if (emp.equals(employeeID)){
       myString = employeeID + " was found.";
       break;
     }
   }
   System.out.println(myString);
   }
```

Reviewing Keywords for Control

```
static Employee[] allEmployees;

static void printManagedBy(String managerID){
 for (Employee emp : allEmployees){
   if (!emp.isManagedBy(managerID)){
    continue;
   }
 System.out.println(emp.getName());
 }
 }
```

Reviewing Keywords for Control

```
public class BreakLoop {

public static void main(String[] args) {

outer: // new label
for (int i = 0; i < 3; i++) {// outer loop (i loop)
 for (int j = 0; j < 3; j++) { // inner loop (j loop)
   System.out.println("i = " + i + " and j = " + j);
   if (i + j == 3) { // same conditional expression
     System.out.println("Break out of both loops.\n");
     break outer; // new break statement with label
   }
}}}}
```

Output:

i = 0 and j = 0
i = 0 and j = 1
i = 0 and j = 2
i = 1 and j = 0
i = 1 and j = 1
i = 1 and j = 2
Break out of both loops.

Conclusions

- Comparisons using Operators and Methods
- Understanding Language Control
- Reviewing Keywords for Control