CONDITIONALS

CITS1001
Scope of this lecture

• Expressions
• Conditional Statements

Source: ppts: Objects First with Java - A Practical Introduction using BlueJ,
© David J. Barnes, Michael Kölling
And Gordon Royle, UWA
MAKING CHOICES
Reflecting on the ticket machines

• Their behavior is inadequate in several ways:
  • No checks on the amounts entered.
  • No refunds.
  • No checks for a sensible initialization.

• How can we do better?
  • We need more sophisticated behavior.
Making choices in everyday life

• If I have enough money left, then I will go out for a meal
• otherwise I will stay home and watch a movie.
Making a choice in everyday life

if(I have enough money left) {
    go out for a meal;
}
else {
    stay home and watch a movie;
}
Making choices in Java

`if` keyword

boolean condition to be tested

`perform some test`

actions if condition is true

`Do these statements if the test gave a true result`

`else`

actions if condition is false

`Do these statements if the test gave a false result`

`else` keyword
Making a choice in the ticket machine

```java
public void insertMoney(int amount)
{
    if(amount > 0) {
        balance = balance + amount;
    }
    else {
        System.out.println(
            "Use a positive amount: " + 
            amount);
    }
}
```
How do we write 'refundBalance'?
Variables – a recap

• Fields are one sort of variable.
  • They store values through the life of an object.
  • They are accessible throughout the class.

• Parameters are another sort of variable:
  • They receive values from outside the method.
  • They help a method complete its task.
  • Each call to the method receives a fresh set of values.
  • Parameter values are short lived.
Local variables

• Methods can define their own, *local* variables:
  • Short lived, like parameters.
  • The method sets their values – unlike parameters, they do not receive external values.
  • Used for ‘temporary’ calculation and storage.
  • They exist only as long as the method is being executed.
  • They are only accessible from within the method.
Scope highlighting

```java
/**
 * Print a ticket if enough money has been inserted, and
 * reduce the current balance by the ticket price. Print
 * an error message if more money is required.
 */
public void printTicket()
{
    if (balance >= price) {
        // Simulate the printing of a ticket
        System.out.println("################################################");
        System.out.println("# The Blue Line");
        System.out.println("# Ticket");
        System.out.println("# " + price + " cents.");
        System.out.println("################################################");
        System.out.println();
        // Update the total collected with the price.
        total = total + price;
        // Reduce the balance by the price.
        balance = balance - price;
    }
    else {
        System.out.println("You must insert at least: " +
                        (price - balance) + " more cents.");
    }
}
/**
 * Return the money in the balance.
 * The balance is cleared.
 */
```
Scope and lifetime

• Each block defines a new scope.
  • Class, method and statement.
• Scopes may be nested:
  • statement block inside another block inside a method body inside a class body.
• Scope is static (textual).
• Lifetime is dynamic (runtime).
Local variables

public int refundBalance() {
    int amountToRefund;
    amountToRefund = balance;
    balance = 0;
    return amountToRefund;
}
Scope and lifetime

- The scope of a local variable is the block in which it is declared.
- The lifetime of a local variable is the time of execution of the block in which it is declared.
- The scope of a field is its whole class.
- The lifetime of a field is the lifetime of its containing object.
// This method should return how many of its three arguments are odd numbers.
public static void printNumOdd(int n1, int n2, int n3) {
    int count = 0;
    if (n1 % 2 != 0) {
        count++;
    } else if (n2 % 2 != 0) {
        count++;
    } else if (n3 % 2 != 0) {
        count++;
    }
    System.out.println(count + " of the 3 numbers are odd.");
}
Summary

• ‘Correct’ behavior often requires objects to make decisions
• Objects can make decisions via conditional (if) statements
• A true-or-false test allows one of two courses of actions to be taken
Summary of Conditional Statement Rule

- First evaluate the expression (giving true or false)
- If true then execute the first statement(s)
- Otherwise execute the second statement(s)
- The else part is optional

```java
if (input.contains("bye")) {
    finished = true;
} else {
    response = responder.generateResponse(input);
    System.out.println(response);
}
```