Implementation

CSE 142, Summer 2003

Computer Programming 1

http://www.cs.washington.edu/education/courses/142/03su/

Readings and References

- Reading
  - » Chapter 4, *Intro to Programming and Object-Oriented Design Using Java*, Niño and Hosch
- References
  - » The Acrobat code examples are from the zip file provided for the previous lecture

Outline for Today

- Implementing classes in Java
- Instance variables – properties
- Value-returning methods for queries
- Void methods for commands
- Return statement
- Assignment statement and expressions
- Method parameters
- Constructors

Specification vs Implementation

- Specification – external view of an object/class
  - » View of the class as seen by *client* code (i.e., other code that creates or uses objects of this class)
  - » Class name and method names, parameters, and descriptions
- Implementation – internal details private to the class
  - » Instance variables – properties
  - » Statements that define the methods

Instance variables store the properties

```
/** the surname or family name of this person */
private String familyName;
/** the cumulative action count for this Acrobat. */
private int actionCount;
```

- These are *instance variable* declarations

  `private <type> <identifier>;`

  - » private – these properties are part of the Acrobat implementation, not visible to other kinds of objects
  - » <type> - the type of the variable
  - » <identifier> - a meaningful name for the variable
- Each object of class Acrobat has its own copy of this list of instance variables

Implementing Methods for Simple Queries

```
/**
* Tell the caller what our given name is.
* @return the given name
*/
public String getGivenName() {
    return givenName;
}
```

- When this method is executed, it replies with the value of the instance variable givenName

```
Acrobat me = new Acrobat("Doug", "Johnson");
String who = me.getGivenName();
System.out.println(who+" is showing this example.");
```


Value-Returning (Query) Methods

- Form

```
/** Comment specifying the method */
public  <result type>  <identifier> ( ) {
      list of statements
}
```

- Details
  - » public – this method is part of the public specification of the class
  - » <result type> – the type of the value returned by this query
  - » <identifier> – the name of this method
  - » *list of statements* – the *body* of the method
    - These make up the algorithm that the method executes when it is called

Return Statement

- First example of a statement

  `return` ***expression*** `;`
- Meaning
  - » Evaluate the expression to get a value
    - In **`getActionCount`**, the expression is the value of the instance variable **`actionCount`**
    - For a variable, evaluation means get its current value
  - » Then, finish execution of this method (query), replying with the value of the expression

Arithmetic Expressions

- Basic components of an expression
  - » Literals – 17, 3.0, 1.023e23, true, "Hi there", 'k'
    - ints will be upgraded to double when appropriate
  - » Variable names – value is the current value of the variable
- Operators (see book for all the details)
  - » +, -, *, /, % (remainder)
    - Gotchas: for ints, x/y yields integer part, dropping any fraction; x%y gives the remainder
  - » Operators have the usual *precedence*
    - For example, a + b * c is understood to mean a + (b * c)
    - Use parentheses to make it clear what you mean (a + b) * c

Methods for Simple Commands

```
/**
* Clap as instructed.  No fancy audio capabilities, so our
* clapping is limited to just saying that we clapped.
* @param k the number of times to clap
*/
public void clap(int k) {
 System.out.println(familyName+" clapped "+k+" times.");
 actionCount = actionCount + k;
}
```

- When this method is executed, what does it do?

```
me.clap(2);
```


Command (Action) Methods

- Form

```
/** Comment specifying the method */
public  void  <identifier> ( parameters ) {
       list of statements
}
```

- Details
  - » public, <identifier>, and *list of statements* – same as for queries
  - » void – this command method doesn't return a value
  - » parameters – information supplied to the command
    - Same form as a variable declaration
    - Queries can also have parameters, but they have not been needed in the simple cases we've seen so far

Assignment Statement

- Second example of a statement

  variable = *expression* ;
- Meaning
  - » First, evaluate the expression to get a value
  - » Second, bind that value to the *variable* whose name appears on the left
  - » These two steps are done in that order, not simultaneously
  - » Question: what does this mean (or do)?

```
count = count + 1;
```

Constructor

```
/**
 * Create a new Acrobat using the name information provided.
 * @param given the specific name of this person
 * @param family the surname or family name of this person
 */
public Acrobat(String given,String family) {
 givenName = given;
 familyName = family;
 actionCount = 0;
}
```

- A constructor is executed each time a new Acrobat instance is created
  - » A constructor initializes newly created objects to some sensible state
  - » Syntax difference from other methods - no result type because the result is always a new object

Summary

- Implementation of classes and object
  - » Instance variables
    - hold the current value of a property for this object
    - defined using a type plus a name in class definition
  - » Methods
    - implement the responsibilities of a class
    - defined using a return type, a name, and statements that make up the body of each method
  - » Constructors
    - new objects are created with **`new ClassName(…)`**
    - the constructor fills in the details of the new object