Java CPD (II)

Frans Coenen

Department of Computer Science

Tea Time?

UNIVERSITY OF LIVERPOOL

Java “Keyboard” Input

Keyboard Input (1)

- Java is a sophisticated language that allows input from many streams, the “keyboard” is only one of these.
- Because of this sophistication Java keyboard input is more complicated than with respect to many other languages.
- We read keyboard input using a method called `nextLine()` which is in the `Scanner` class that comes with Java but is not automatically included (unlike some other classes).

Keyboard Input (2)

- Therefore we have to import the `Scanner` class using an `import` statement:

```
import java.util.*;
```

- The `Scanner` class is contained in the "*package*" `Util` which is a subclass of `Java` (everything is a subclass of `Java`)

Keyboard Input (3)

- To use the nextLine method we need to create an instance of the class Scanner:

```
private static Scanner keyboardInput =
          new Scanner(InputStream source);
```

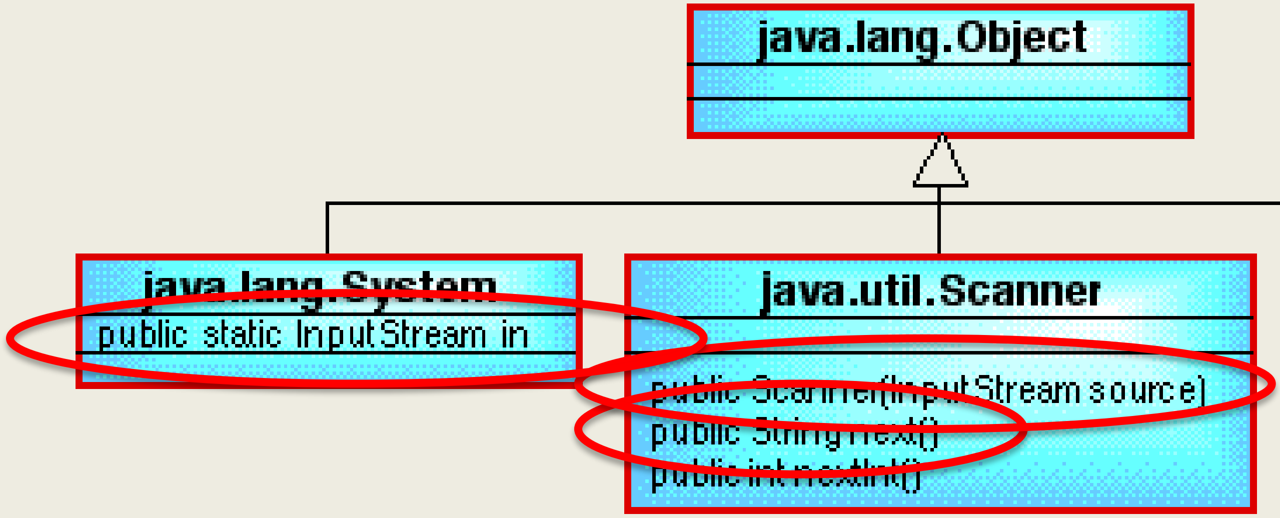
- Static because we do not want to (accidently) change it. The argument is an instance of the class `InputStream` class called `source`.
- Java supplies an appropriate instance of the class `InputStream` in the `System` class (its called `in`).

Keyboard Input (4)

- Keyboard input, by default, is always in the form of a string, thus if we wish to input a double or an integer we must specify this:

```
keyboardInput.nextDouble();
keyboardInput.nextInt();
```

Keyboard Input (5)



```
private static Scanner keyboardInput =
      new Scanner(InputStream source);
String s = keyboardInput.next();
double d = keyboardInput.nextDouble();
```

Problem Example 5: Landscape Gardening Quote With Keyboard Input

Landscape Gardening Quote With Keyboard Input Requirements*

Create a Java programme that: (a) allows a user to input lawn and patio dimensions and the number of water features required (if any); and (b) outputs the cost for each.

* (Taken form AQA HCSE Specimen Controlled Assessment v1.0)

Quote Item Source Code (ver. 4)

- Go to the directory `H:\JavaCPD\JavaExampleProgrammes\keyboardInput\LandscapeGardKBinput` and load `QuoteItem.java` into the text editor.
- Note that we have imported the package `Java.util`.
- Note that we have created an instance of the `Scanner` class (protected so that it can be inherited).

```
protected static Scanner input =
                    new Scanner(System.in);
```

Quote Item Type 1 and 2 Source Code (Ver. 3)

- Load `QuoteItemType1.java` (`QuoteItemType2.java` into the text editor.

- **Note**: We have added another constructor in each case with just three arguments no `length` or `width` (`quantity`) with keyboard input for `length` and `width` (`quanitity`).

Compiling and Running The Quote Item Application

```
javac *.java
```

```
java QuoteItemApp
```

- Try adding another feature such as a patio or garden lights (remember to also add an appropriate output statement).

Constants

Constants

- Constants are data items whose value cannot be changed (it is “constant”)
- In Java, by convention, we indicate that a data item is constant by using all upper case for the name.
- Thus:

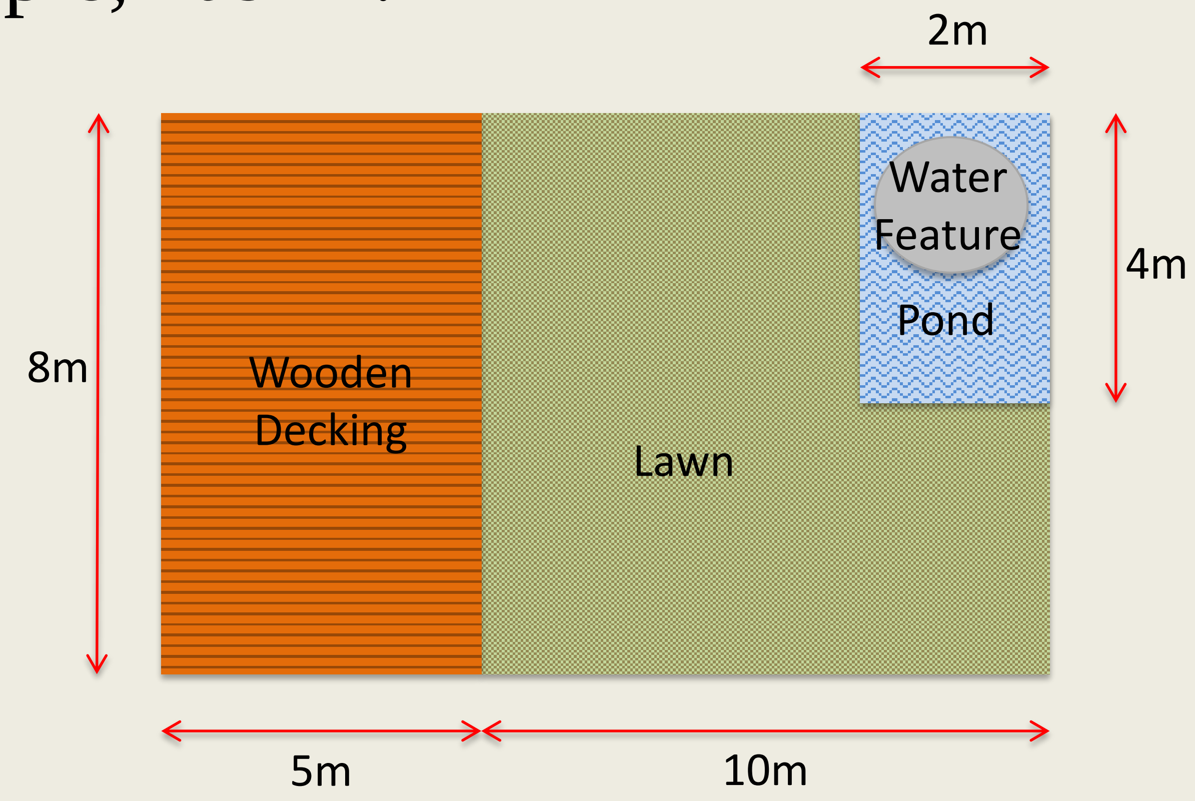
```
private static double
                INST_TIME_LAWN = 20.0;
```

Problem Example 6: Landscape Gardening Task 1(a) (The Full Quote)

UNIVERSITY OF LIVERPOOL

Landscape Gardening Task 1(a)

- AQA GCSE Specimen Controlled Assessment example, Task 1.



Landscape Gard. Task 1(a) Reqs.

- (Taken from AQA GCSE specimen). Customers provide a landscape gardening company with a plan. Costs are as shown in the table. There is also a labour charge of £16.49 for every hour of work done. Create a java programme that: (a) allows a user to input lawn and patio dimensions and the number of water features required (if any); and (b) outputs the cost for each, the labour cost and the total cost.

Work to be done	Cost of materials	Time to install
Laying a lawn	£15.50 per m^2	20 mins per m^2
Laying a concrete patio	£20.99 per m^2	20 mins per m^2
Installing a water feature (e.g. a fountain)	£150.00 each	60 mins each

Quote Source Code

- Go to the directory `H:\JavaCPD\JavaExampleProgrammes\KeyboardInput\LandscapeGardeningTask1a` and load `Quote.java` into the text editor.

- Note

1. New class. Has fields for: (i) the month, (ii) instances of `QuoteItemType1` (two of these) and `QuoteItemType2` (one of these), (iii) labour cost and (iv) various totals.

Landscape Gardening Source Code

- Load `LandsGardQuote.java` into the text editor.

- Note

1. Another new class.
2. Constants for installation times and labour costs.
3. Fields for: (i) instance of `Scanner` class, (ii) material costs and (iii) an instance of `Quote`.
4. Method `prepareAnewQuote().`

Landscape Gardening Application Source Code

- Load `LandsGardQuoteApp.java` into the text editor.

- The application class has also been revised.

Compiling and Running The Quote Item Application

```
javac *.java
```

```
java QuoteItemApp
```

- Try adding another landscape gardening element, for example garden lights.

EditingThe Quote Item Application

In `LandsGardQuote` class add:

1. Material cost constant
2. Installation time constant
3. In `inputQuoteDetail()` method add:
   - Line to create quote element
   - argument for new item to `Quote` constructor call

In `Quote` class add

1. Field for new feature
2. Argument and assignment in Quote constructor
3. In `caluateCost` method update:
   - `totalInstallationTime` calculation
   - `totalMaterialCost` calculation
4. Edit `toString` method to include new feature

Programme Constructs

Programme Constructs

- Programming is founded on three basic constructs:

    1. Sequence
    2. Selection
    3. Repetition

Selection (linear “if” and “if-else”)

Linear “if”

```
if ( <BOOLEAN_EXPRESSION> {
   <STATEMENTS>
   }
```

“if-else”

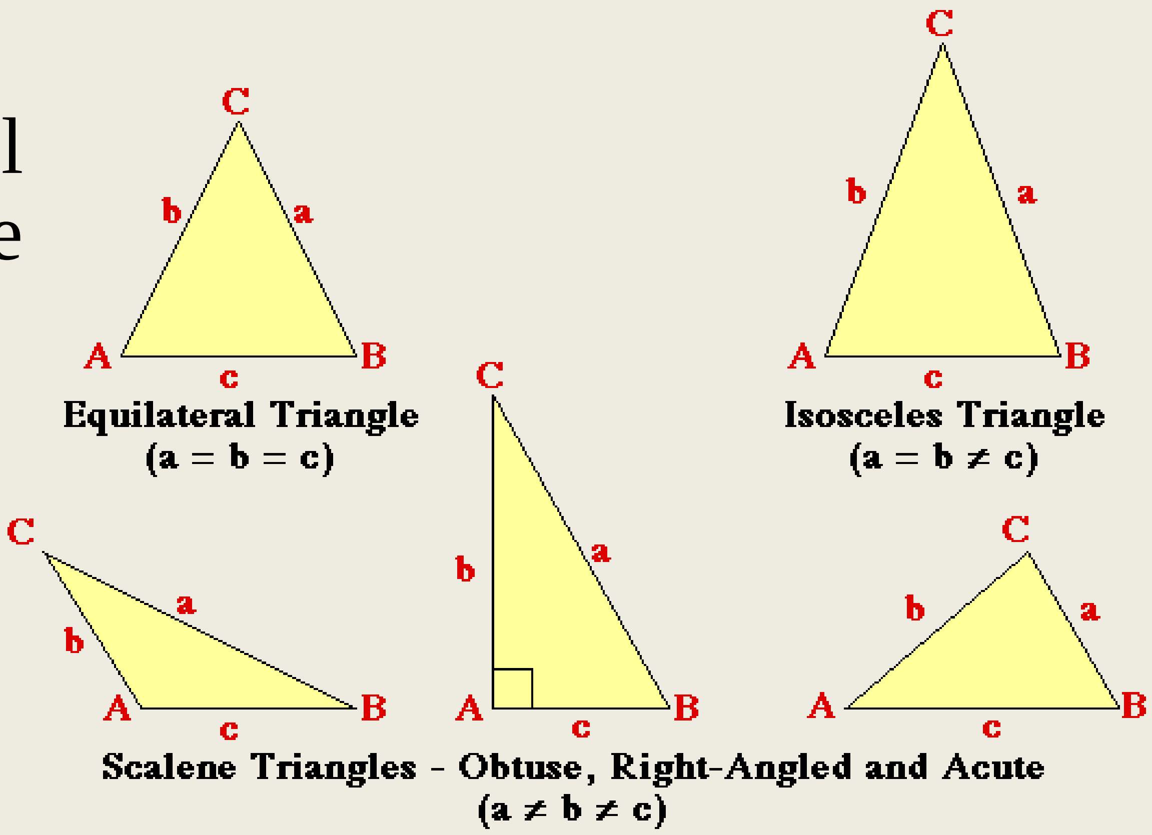
```
if ( <BOOLEAN_EXPRESSION> {
   <STATEMENTS_1>
   }
else {
   <STATEMENTS_1>
   }
```

Problem Example 7: Triangle Recognition

Triangles Requirements

- Produce a Java program which, given three sides of a triangle, determines whether the triangle is either:

1. Equilateral (all sides the same length),
2. Isosceles (two sides the same length), or
3. Scalene (no sides the same length).



Triangle Source Code

- Go to the directory `H:\JavaCPD\JavaExampleProblems\Selection\TriangleRecognotion` and load `TriangleRecog.java` into the text editor.

- Note: Includes test to determine whether input triangle can be realised or not.

- Load `TriangleRecogApp.java` into the text editor.

Compiling and Running The Triangle Recognition Application

```
javac *.java
```

```
java TriangleRecogApp
```

Switch Statements

UNIVERSITY OF LIVERPOOL

Switch Statement

```
Switch (<SELECTOR>) {
    case <VALUE_1>:
        <STATEMENTS_1>
    case <VALUE_2>:
        <STATEMENTS_2>
    default:
        <DEFAULT_STATEMENTS>
        }
```

- Used to select between a number (more than two) of alternatives.

While Loops

The While Loop Statement

```
While (<CONDITION>{
    <STATEMENTS_TO_BE_REPEATED>
    }
```

- General purpose loop for repeating some sequence of statements.

Problem Example 8: Menu Input

Menu Input Requirements

Design and implement a Java application class that allows the user to select from five different menu options on a continuous loop, including an option to quit the program. Include an error handling mechanism for the situation where an unrecognised menu option is input by the user.

Menu Application Source Code

- Go to the directory `H:\JavaCPD\JavaExampleProblems\Repetition\MenuApp` and load `MenuApp.java` into the text editor.

- Note:

1. Note that source code features an infinite loop (the test statement comprises the Boolean vale `True` which evaluates to itself) and that the termination statement is embedded in the loop using an "if" statement.
2. A switch statement is used to implement the menu (default case at the end).

Compiling and Running The Menu Application

```
javac *.java
```

```
java menuApp
```

Arrays

Arrays (Reminder 1)

- The available primitive (basic) types can be extended by adding compound types made up of existing primitive types (and/or other compound types).
- Compound types are usually programmer defined.
- Classes in java can be viewed as compound types.
- The most straight forward (and oldest) form of compound data type is the array.
- An array can be viewed simply as a collection of data items all of the same type.

Arrays (Reminder 2)

- Features of arrays:

1. Items in arrays are called *elements* (some authors call them cells).
2. The number of elements in an array is described by its *length*.
3. Specific elements in an array can be identified through the use of an *index*.
4. In Java the index is always of the type integer (this is not the case in all programming languages).
5. In Java the first index is always the integer `0` which is referred to as the *lower bound*, the last index is then referred to as the *upper bound*.
6. Note that: `upperBound = length-1.`

For Loops

The For Loop Statement

```
for (<StartExpression> ; <TestExpression> ;
                         <UpdatExpression>) {
   < sequence of statements >
   }
```

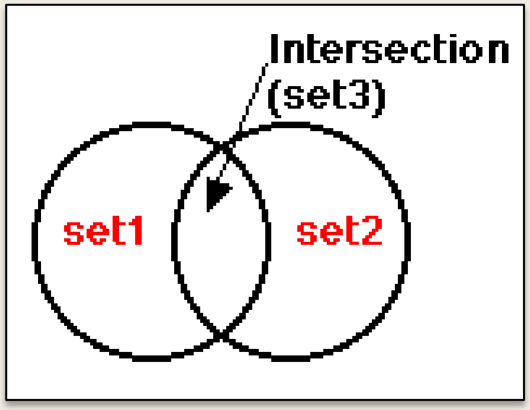
- **Start expression**: Introduces one or more loop parameters (also referred to as the control variables or loop counters)
- **Test expression**: Defines the end condition for terminating the loop.
- **Update expression**: To avoid infinite repetition the loop parameter(s) must be updated on each repetition.

Problem Example 9: Set Intersection

Set Intersection Requirements

Develop a Java program which, given two sets of integers, determines the intersection of these two sets and stores this in a third set. For example Given:

set1 = {2 4 6 8 10 12 14 18 20}
set2 = {3 6 9 12 15 18}



set1 intersection set2 = set3 = {6 12 18}

Set Intersection Source Code

- Go to the directory `H:\JavaCPD\JavaExampleProgrammes\Arrays\SetIntersection` and load `SetOperations.java` into the text editor.

Set Intersection Comments (1)

- The set is defined as an integer array (line 16) whose size is specified by the constructor (line 21).
- The `noDuplicates` method (lines 47-57) includes a straightforward for loop (by definition sets cannot contain duplicate items).
- The `toString` method (lines 155-169) also includes a for loop (note index set to `1` not zero because first element has already been considered).

Set Intersection Comments (2)

- The `inputSet` method (lines 28-40) includes a for loop with the update expression embedded in the loop.
- The `intersection` method (lines 65-90) contains two for loops, one nested inside the other.
- The `numIntersectingElements` method (lines 96-120) also contains two for loops, one nested inside the other.

Set Intersection Application Source Code

- Load `SetIntersectionApp.java` into the text editor.

- **Note**: We create two sets (instances of the `SetOperations` class) and then find the intersection between these two sets.

Compiling and Running The Set Intersection Application

```
javac *.java
```

```
java SetIntersectionApp
```

- Run the application a few times creating different pairs of sets (include empty sets of length zero).
- Try creating anther set, `set4`, and finding the intersection between this and `set3` (the intersection of `set1` and `set2`).

Home Time?

UNIVERSITY OF LIVERPOOL