Review of basic Java 1

(contains some material from slides accompanying Horstmann: Java for Everyone: Late Objects, John Wiley and Sons Inc)

CSCU9A2 Review of basic Java 1
© University of Stirling 2017

1

Overview

- Review of basic Java:
  - Variables
  - Values
  - Assignment statements
  - Expressions
- With a focus on:
  - Formal syntax definition
  - Compiling schemes

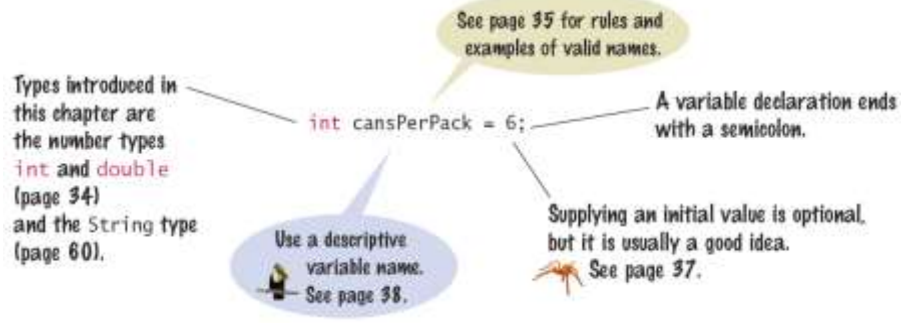
CSCU9A2 Review of basic Java 1
© University of Stirling 2017

2

Variable Declarations

A variable is a storage location with a name

- When declaring a variable, you tell the compiler the *type* of data it will hold, optionally you can specify an initial value


See page 35 for rules and examples of valid names.
Types introduced in this chapter are the number types int and double (page 34) and the String type (page 60).
int cansPerPack = 6;
A variable declaration ends with a semicolon.
Use a descriptive variable name. See page 38.
Supplying an initial value is optional, but it is usually a good idea. See page 37.

CSCU9A2 Review of basic Java 1
© University of Stirling 2017 3

What can a variable name be?

- Reference "syntax rules" from:
  http://docs.oracle.com/javase/specs/jls/se7/html/index.html :

  Identifier:
  IdentifierChars
  but not a Keyword *or* BooleanLiteral *or* NullLiteral

  IdentifierChars:
  JavaLetter
  IdentifierChars JavaLetterOrDigit

  JavaLetter:
  any Unicode character that is a Java letter

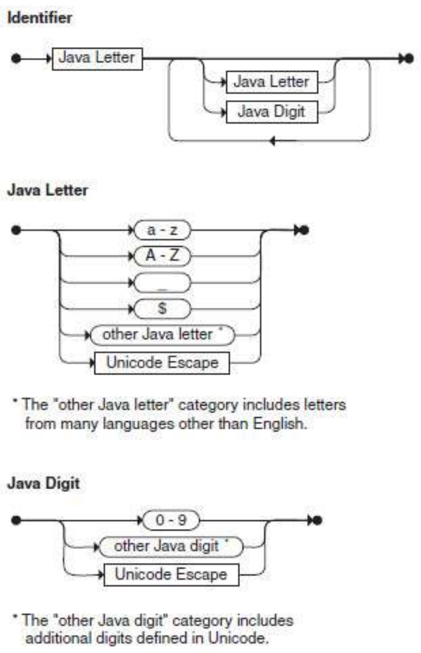
  JavaLetterOrDigit:
  any Unicode character that is a Java letter-or-digit

- "Backus–Naur Form" (BNF) "rewriting rules"
  - Invented around 1959

CSCU9A2 Review of basic Java 1
© University of Stirling 2017 4

- Or "syntax diagrams" or "railroad diagrams" from *http://markettorrent.com/ topic/9359* *(a simplified view of Java)*
- Follow the railroad tracks...
- One of the first uses was to define Pascal in 1973

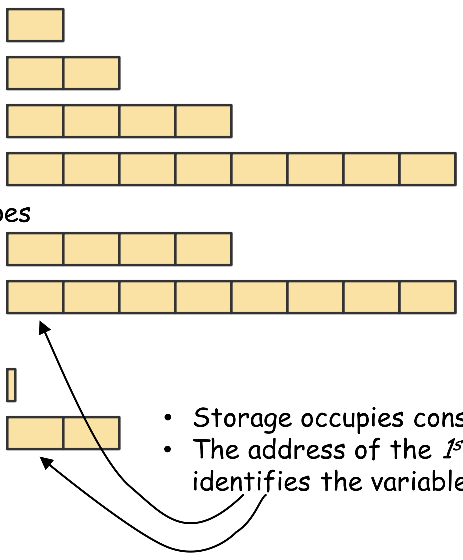
- (Where these differ from the "reference rules" then the "reference rules" are definitive)



Identifier
Java Letter
Java Letter
Java Digit
Java Letter
a - z
A - Z
_
$
other Java letter *
Unicode Escape
* The "other Java letter" category includes letters from many languages other than English.
Java Digit
0 - 9
other Java digit *
Unicode Escape
* The "other Java digit" category includes additional digits defined in Unicode.

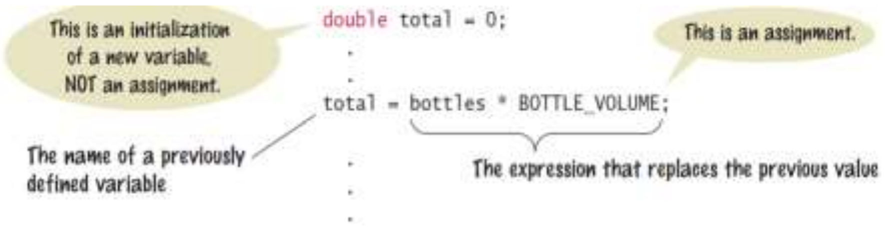
Variable Storage per Type (in bytes)

- Integer Types
  - `byte:`
  - `short:`
  - `int:`
  - `long:`
- Floating Point Types
  - `float:`
  - `double:`
- Other Types
  - `boolean:`
  - `char:`



- Storage occupies consecutive bytes
- The address of the *1st byte* identifies the variable's storage

Assignment Statement Syntax


This is an initialization of a new variable, NOT an assignment.
double total = 0;
This is an assignment.
total = bottles * BOTTLE_VOLUME;
The name of a previously defined variable
The expression that replaces the previous value

- The value on the right of the '=' sign is copied to the variable on the left
- More correctly:
  - The value computed from the *expression* on the right of the '=' is copied into the memory storage location for the variable on the left, starting at its first byte

CSCU9A2 Review of basic Java 1
© University of Stirling 2017

7

Assignment Statement Syntax

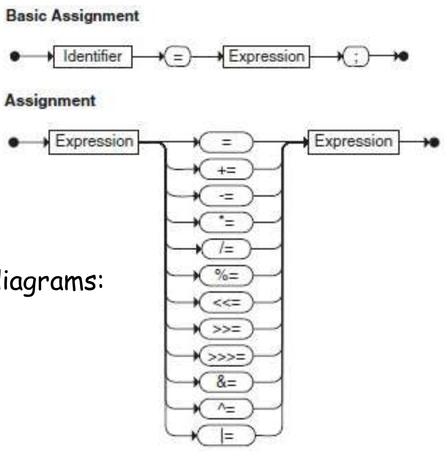
- Reference syntax rules:

  Assignment:
  LeftHandSide AssignmentOperator AssignmentExpression

  LeftHandSide:
  ExpressionName
  FieldAccess
  ArrayAccess

  AssignmentOperator:
  one of = *= /= %= += -=
  <<= >>= >>>= &= ^= |=

- Railroad diagrams:



CSCU9A2 Review of basic Java 1
© University of Stirling 2017

How an assignment statement works

- An assignment statement

  `v = exp;`

- Compiles into machine code – two steps:

  … compile **`exp`** with result to **`Rn`** …
  `MOV Rn -> [v]`

  where **`v`** is now the memory address of variable **`v`**

- Example:

```
MOV  [a] -> R1
MOV  [b] -> R2        for      a = a + b;
ADDI R1, R2 -> R3
MOV R3 -> [a]
```

(pretending that Java compiles to Brookshear machine code)

More about expressions

- Expressions are typically calculations such as:

  `(a + 3) * b`

- The simplest expressions are variables and values ("literals" or "constants")
- `+` and `*` are call *binary operators*
  - Because in expressions they have *two operands:*

    …some expr… + …some expr…

- The formal syntax description of expressions reflects this *recursive* structure
  - Wikipedia has a good definition:
    " *Recursion* is the process of repeating items in a self-similar way. "

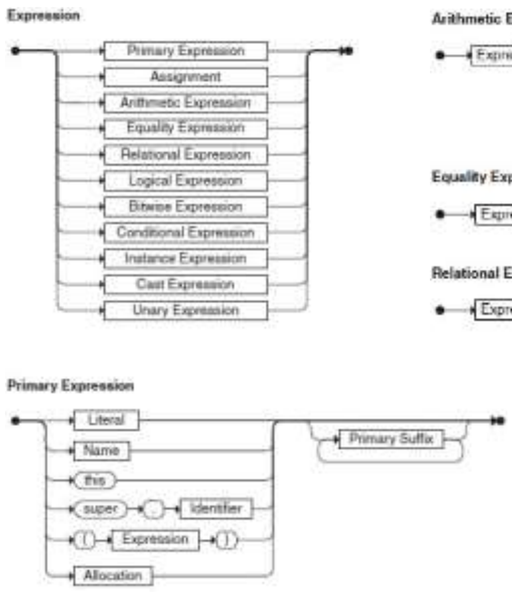
Syntax of expressions

- Reference syntax rules: (extract – there is much more!)

```
AdditiveExpression:
        MultiplicativeExpression
        AdditiveExpression  +  MultiplicativeExpression
        AdditiveExpression  -  MultiplicativeExpression

MultiplicativeExpression:
        UnaryExpression
        MultiplicativeExpression  *  UnaryExpression
        MultiplicativeExpression  /  UnaryExpression
        MultiplicativeExpression  %  UnaryExpression

UnaryExpression:
        + UnaryExpression
        - UnaryExpression
        UnaryExpressionNotPlusMinus
```

```
UnaryExpressionNotPlusMinus :        (simplified)
        Literal
        ExpressionName

Literal:
        IntegerLiteral
        FloatingPointLiteral
        BooleanLiteral
        CharacterLiteral
        StringLiteral
        NullLiteral
```

- Railroad diagrams:



CSCU9A2 Review of basic Java 1
© University of Stirling 2017

13

How an expression is evaluated/compiled

- A variable or literal cause a value to be loaded into a register:

```
MOV [a] -> Rn        for  a    (a is variable/address)
MOV i -> Rn          for  i
```

- Consider simple binary operator expressions:

  …expr A… + …expr B…

- Compiles into machine code, with result in some register **Rc**:

  … compile **expr A** with result to some **Ra** …
  … compile **expr B** with result to some **Rb** …
  ADDI Ra, Rb -> Rc

- Example:

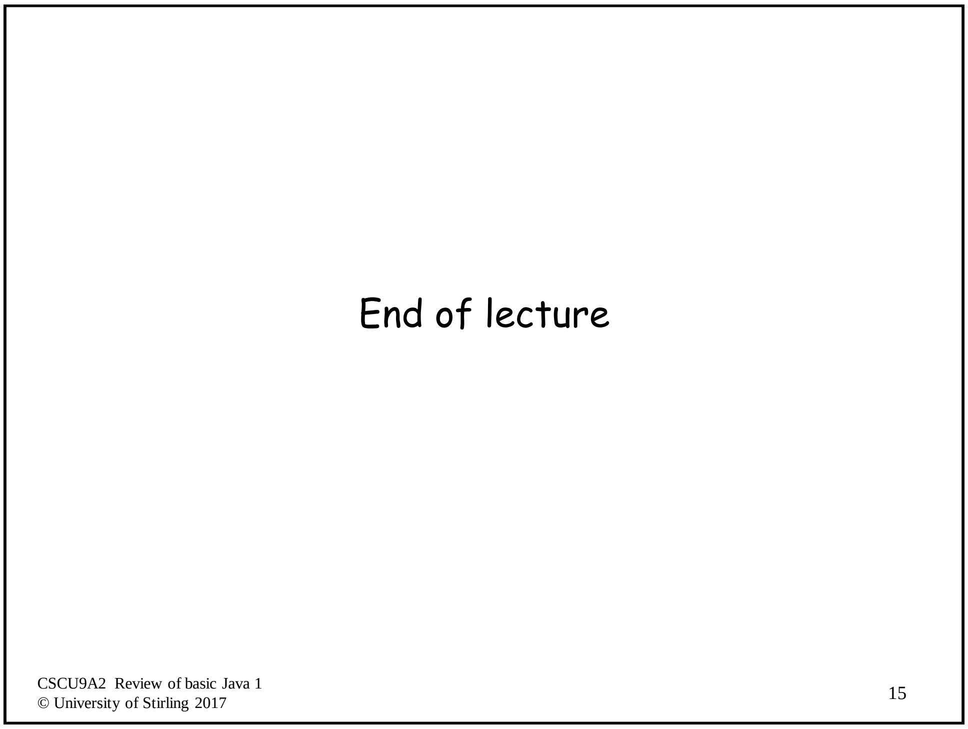
```
MOV [a] -> R1
MOV 03 -> R2
ADDI R1, R2 -> R3
MOV [b] -> R1
MULI R1, R3 -> R2
```

for (a + 3) * b
with result in **R2**

(pretend Brookshear has MULI)

CSCU9A2 Review of basic Java 1
© University of Stirling 2017

14



End of lecture

CSCU9A2 Review of basic Java 1
© University of Stirling 2017

15