2 Language Basics – Variables

- Variables have a type
  - Primitive type
  - Reference type

<table>
<thead>
<tr>
<th>Definition</th>
<th>Declaration</th>
<th>Instantiation</th>
<th>Manipulation</th>
<th>Equality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitive</td>
<td>int a;</td>
<td>a = 117;</td>
<td>a = b + 42;</td>
<td>a == b;</td>
</tr>
<tr>
<td>Reference</td>
<td>class Student { // Fields and // methods ... }</td>
<td>Student heiner;</td>
<td>heiner = new Student();</td>
<td>heiner.age = 21; heiner.yawn();</td>
</tr>
</tbody>
</table>

2 Language Basics – Variables

- Primitive types (numeric, boolean, character):

<table>
<thead>
<tr>
<th></th>
<th>byte</th>
<th>short</th>
<th>int</th>
<th>long</th>
<th>float</th>
<th>double</th>
<th>boolean</th>
<th>char</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8 bit</td>
<td>16 bit</td>
<td>32 bit</td>
<td>64 bit</td>
<td>32 bit</td>
<td>64 bit</td>
<td>1 bit</td>
<td>16 bit</td>
</tr>
<tr>
<td></td>
<td>{2, [2] } {128,127}</td>
<td>{2&lt;sup&gt;31&lt;/sup&gt;-1} {32768,32767}</td>
<td>{2&lt;sup&gt;31&lt;/sup&gt;-2} {32768,32767}</td>
<td>{2&lt;sup&gt;31&lt;/sup&gt;-1} {32768,32767}</td>
<td>{1.4<em>10&lt;sup&gt;-40&lt;/sup&gt; } {1.4</em>10&lt;sup&gt;-38&lt;/sup&gt; }</td>
<td>{-1.8<em>10&lt;sup&gt;30&lt;/sup&gt; } {1.8</em>10&lt;sup&gt;30&lt;/sup&gt; }</td>
<td>{true, false}</td>
<td>{0, ..., 255}</td>
</tr>
</tbody>
</table>

Unicode (UTF-16)

```
bike1 = new Bicycle();
bike2 = new Bicycle();
```

```
boolean c;
c = bike1.equals(bike2); // c == true
c = (bike1 == bike2); // c == false
```
2 Language Basics – Variables

Reference Type Variables

- Reference type variables "point" to an object of the reference type

```
bike1 = new Bicycle();
bike2 = new Bicycle();

bike1.gear = 3;
```

```
boolean c;
c = bike1.equals(bike2);
// c = false
c = (bike1 == bike2);
// c = false
```

Arrays

- Array: "Indexed list" of elements
- Holds a fixed number of variables of a certain type (primitive or reference)
- Is itself a reference type (see next slide)

```
int[] someArray;
someArray[0] = 23;
someArray[1] = 12;
someArray[5] = 4 + someArray[2];

String[] someOtherArray;
someOtherArray[0] = new String[30];
someOtherArray[17] = "bla bla";

AnyClass[] thirdArray;
thirdArray[44] = new AnyClass();
thirdArray[44].someMethod();
```
2 Language Basics – Variables

Arrays

- Array is itself a reference type:

```java
int[] someArray = new int[3];
int[] anotherArray = new int[3];
someArray[2] = 7;
anotherArray[1] = 8;
```

```java
int[] someArray = new int[3];
int[] anotherArray = new int[3];
someArray[2] = 7;
anotherArray[1] = 8;

boolean b = (someArray[1] == 8); // b == true
```

- Array is itself a reference type:

```java
memory [simplified model]

<table>
<thead>
<tr>
<th>cell nr</th>
<th>cell name</th>
<th>cell content</th>
</tr>
</thead>
<tbody>
<tr>
<td>1149</td>
<td>someArray</td>
<td>&lt;1328&gt;</td>
</tr>
<tr>
<td>1150</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1151</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1152</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>1327</td>
<td>anotherArray</td>
<td>&lt;1328&gt;</td>
</tr>
<tr>
<td>1328</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1329</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>1330</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
```

- Length property:

```java
int l = someArray.length; // l == 3
```
2 Language Basics – Operators

Operators

• Operators (mostly) act on variables of primitive types. Examples:

Assignment Operator

= Simple assignment operator (also for reference types)   a = b+1; bike2 = bike1.copy();

Arithmetic Operators

+ Additive operator   double sss = b + 1.7; int a = 1 + 1;
− Subtraction operator   int b = c - 9; float f = 10.0f - 23.0f;
* Multiplication operator   fd = fd * 0.1f; double d = z * z;
/ Division operator   int a = 17 / 9; // a = 1;
\ Remainder operator   float eee = 13.0f / 2.0f; // eee = 6.5f;
%

Unary Operators

+ Unary plus operator; (not very useful)   int a = -1; int b = +a; // b == -1
− Unary minus operator; negates an expression   int a = -1; int b = -a; // b == 1
++ Increment by 1   int a = 0; a++; // a == 1;
-- Decrement by 1   int a = 1; a--; // a == 0;
!

Important: Dereference operator for reference types: dot-operator "."

String s1 = s1.concatenate(s2);
bike1.cadence = 4;
bike1.changeGear(5);

Equality and Relational Operators

== Equal to   boolean a = (1 == 1); // a == true
!= Not equal to   boolean a = (1 != 1); // a == false
> Greater than   boolean a = (17 > 12); // a == true;
>= Greater than or equal to   etc.
< Less than
<= Less than or equal to

Conditional Operators

&& Conditional AND   a = false; b = true; c = a && b; // c == false;
|| Conditional OR   a = false; b = true; c = a || b; // c == true;
?

?: Ternary (shorthand for if-then-else statement, use if-then-else instead!)

Reference Type Comparison Operator

instanceof Compares an object to a specified type   Vector z = new Vector();
boolean b = z instanceof Vector; // b == true;

Bitwise and Bit Shift Operators

(not that important for us; see URL below)

http://java.sun.com/j2se/1.3/docs/api/java/lang/Boolean.html

Expressions

• Expression: Legal combination of constants, variables and operators
• Can be (and typically are) nested
• Expressions evaluate to a value of a certain type

Given:   int a = 73;   boolean someArray[] = new boolean[5];

<table>
<thead>
<tr>
<th>Example</th>
<th>Evaluates to</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>48</td>
<td>int</td>
</tr>
<tr>
<td>2.0 / 3.0</td>
<td>0.66666666666</td>
<td>double</td>
</tr>
<tr>
<td>true</td>
<td></td>
<td>boolean</td>
</tr>
<tr>
<td>15 / 8</td>
<td>1</td>
<td>int</td>
</tr>
<tr>
<td>(17 + (3 * 9)) % 3</td>
<td>2</td>
<td>int</td>
</tr>
<tr>
<td>a + 1</td>
<td>74</td>
<td>int</td>
</tr>
<tr>
<td>a * 9.0 / someArray.length</td>
<td>131.4</td>
<td>double</td>
</tr>
</tbody>
</table>
2 Language Basics – Expressions, Statements, Blocks

Expressions

- Expression: Legal combination of data
- Can be (and typically are) nested
- Expressions evaluate to a value

Given: `int a = 73; int b;`

<table>
<thead>
<tr>
<th>Example</th>
<th>Value</th>
<th>Side-effect</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>a = 84</code></td>
<td>84</td>
<td>Assign 84 to a</td>
</tr>
<tr>
<td><code>b = (a = 48)</code></td>
<td>48</td>
<td>Assign 48 to both a and b</td>
</tr>
<tr>
<td><code>a++</code></td>
<td>(a)</td>
<td>Assign 49 to a (l)</td>
</tr>
<tr>
<td><code>++a</code></td>
<td>(a)</td>
<td>Assign 50 to a (l)</td>
</tr>
<tr>
<td><code>new Bicycle()</code></td>
<td>Reference to a new instance of Bicycle, e.g. &lt;1150&gt;</td>
<td>Create and initialize new instance of class Bicycle in memory</td>
</tr>
<tr>
<td><code>new double[10]</code></td>
<td>Reference to a new array of double</td>
<td>Create and initialize new array in memory</td>
</tr>
</tbody>
</table>

Statements

- Statement: Complete unit of execution (ends with ";")
  - Assignment expressions: `a = (17 + (3 * 9)) % 3;`
  - Use of ++ or --: `a++;`
  - Method invocations: `someObject.methodOne();`
  - Object creation expressions: `new SomeClass();`
  - Declaration statements: `int a = 0;`
  - Blocks
    - (next slide)
    - Control flow statements
      - (later)
2 Language Basics – Expressions, Statements, Blocks

Blocks

- **Block**: Group of zero or more statements enclosed in "{" ... "}"

```java
if (a == b) {
    c = 17;
    f++;
    bbb.someMethod();
} // end block
```

- Variables declared inside a block are only visible from within that block:

```java
int a = 7, b = 6;
if (a != b) {
    int c;
    c = a * b;
    System.out.println(c);
} // end block
System.out.println(c); // ERROR: c unavailable
```

2 Language Basics – Control Flow Statements

Control Flow Statements

- **Control flow statements**: Allow for deviation of control flow from sequential order of statements:
  - conditionals: if, if else, switch
  - loops: while, do while, for
  - branches: break, continue, return

```java
void applyBrakes(){
    if (speed > 0) {
        speed = speed - 1;
    }
}
```

```java
void applyBrakes(){
    if (speed > 0) {
        speed--;
    } else {
        System.err.println(
            "The bicycle has already stopped!");
    }
}
```

- **switch**: Equivalent to sequence of chained if else statements
2 Language Basics – Control Flow Statements

- **while**: do something as long as some condition (boolean expression) is true

  ```java
  int count = 1;
  while (count < 8) {
      System.out.print("#:" + count + " ");
      count++;
  }
  output will be: #:1 #:2 #:3 #:4 #:5 #:6 #:7
  ```

- **do while**: similar to "while", but check condition at the end of execution of something instead of at the beginning

  ```java
  int count = 1;
  do {
      System.out.print("#:" + count + " ");
      count++;
  } while (count < 8)
  output will be: #:1 #:2 #:3 #:4 #:5 #:6 #:7
  ```

- **for**: usually means to do something for a fixed number of times:

  ```java
  for (int i=0; i<7; i++) { // loop will be executed 7 times
      System.out.print("#:" + i + " ");
  }
  output will be: #:0 #:1 #:2 #:3 #:4 #:5 #:6
  ```

  **General form**:

  ```java
  for (initialization; termination; update) {
      statement*
  }
  ```

  - **initialization** expression: Executed once at the beginning of first loop
  - **termination** expression: If true then execute statement(s), else exit loop
  - **update** expression: Executed after each iteration of the loop
2 Language Basics – Control Flow Statements

- **break**: force termination of a loop
- **continue**: skip current iteration of a loop

```java
for (int i=0; i<10; i++) {
    if (i == 8) {
        break;
    } else if (i % 2 == 0) {
        continue;
    }
    System.out.print("#": i + " ");
}
```

Output will be: 

```
#1 #3 #5 #7
```

- **return**: terminate current method and return control flow to where the method was invoked from (will be covered shortly in more detail)

3 Classes, Objects, Inheritance

Deepening readings:

- [Java Sun](http://java.sun.com/docs/books/tutorial/javaOO/classes.html)
- [Java Sun](http://java.sun.com/docs/books/tutorial/javaOO/objects.html)
- [Java Sun](http://java.sun.com/docs/books/tutorial/javaOO/more.html)
- [Java Sun](http://java.sun.com/docs/books/tutorial/java/lang/subclasses.html)
- [Java Sun](http://java.sun.com/docs/books/tutorial/essential/exceptions/index.html)