Review of Previous Lecture

• Describe the two major parts of Objects. Describe the relationship between Class & Object.

• Design a simple class (write out attributes & methods) for Bicycle. Create two objects of the class (by writing out main attribute values).

• Result & type (integer or String) of the expressions below
  – “34” + “56”
  – “34” + 56
  – 34 + 56
  – 34 + ( “5” + 6 )
Today’s contents

• More on Object Oriented Programming

• Primitive Data Types
  – Integer
  – Boolean
  – Float/Double
  – Character
Object Oriented Programming (OOP)

- We talked about the relationship between Class & Object
- One more example on Class / Superclass / Subclass / Object
Object Oriented Programming (OOP)

• Here we briefly talk about how to use existing classes.
• We have already used Classes / Objects in last course

```
Scanner s = new Scanner(System.in);
```

• Objects are created by the “new” keyword.
The syntax:
```
ClassType variableName = new ClassType( parameters .... );
```

• How do I know what parameters to put?
  – Generally, check code documentation
  – For this course, you will be told the exact parameters to use
Object Oriented Programming (OOP)

Scanner s = new Scanner(System.in);

• We have an object of Scanner type in variable s now.
• To invoke object members (attributes / methods), use “.”:

```
  s.next();  <---------- this returns a String
  s.nextInt();  <-------- this returns an integer
  s.useDelimiter( ... );  <------ use a different delimiter in separating input
```

Check Java API for all attributes / methods under each built-in class
Object Oriented Programming (OOP)

```java
Student s = new Student();
s.name = "Alan";
s.contact = "919-xxx-xxxx";
s.program = "biostat";
s.year = "1st";

System.out.println(s.name + "'s program is " + s.program);
System.out.println("He is a " + s.year + " year student");
```

----------------------------------
Alan’s program is biostat
He is a 1st year student
Object Oriented Programming (OOP)

• Just now we assumed that all attributes are of String type
• Attributes can be of any data type (including Objects)
• For example, we can have an attribute called Friend in Student class. The type of Friend is also Student.

```java
Student s1 = new Student();
s1.name = “Alan”;
...
Student s2 = new Student();
s2.name = “Anna”;
...
s1.Friend = s2;
```
Account a = new Account();
a.deposit(100);
a.withdraw(50);
System.out.println("The current balance is" + a.getBalance());

a.withdraw(20);
System.out.println("The current balance is" + a.getBalance());

a.deposit(40);
System.out.println("The current balance is" + a.getBalance());
Variables

Summary of “How to use variables”

• **Declare** a variable
  
  ```
  int i; String name; Scanner s;
  ```

• **Assign** a value to the variable (can be an update)
  
  ```
  i = 6; i = 7; i = i + 1;
  ```
Formal rules to name a variable

• Letters, digits(0-9), underscore (_)
• First character cannot be a digit
• No spaces in variable name
• Java is case sensitive

• Legal names
  – pinkFloyd, the_coup, b3atles
• Illegal names
  – michael.bolton, kenny-G, 1CP

• Avoid keywords (see appendix): if, else, return, new
Data types

– Each variable, expression, attribute, method-return-value has a type

– Class type - objects with both data and methods
  • Name begins with uppercase letter (recommended)
  • Scanner, String (Java built-in)
  • Student (we defined)

– Primitive type - indecomposable values
  • Names begin with lowercase letters
  • int, double, char, boolean ...
  • See Figure 2.1, p 52 for full list
Memory & Data & Variable

- Memory is just a long long list of binary digits (bits) that you can use to store information:

```
00011001 00100101 00110000 10101111 0111
```

Unit Addr: 0 1 2 3 4

The minimum unit we consider is 8bits (1byte). We assign an address to each of the unit. So now it looks like a long street with apartments of unit size. Each apartment can hold equal amount of data (8bit = 256 states)

How many units can we have at most?

32-bit OS -> $2^{32}$ → $2^{32} \times 1\text{byte} = 4\text{GB}$

64-bit OS -> $2^{64}$
Memory & Data & Variable

00011001001001010011000010101111011

Unit Addr: 0 1 2 3 4

• Variable is just the unit address – corresponds to a location in memory
• Data type determines how many consecutive units it occupies
• After a variable is declared, it prevents the units from being used by others

• Integer: 32bit -> 4 bytes -> occupies 4 memory units here
• Object: the length is variable. but in units of bytes.
Primitive Types

00011001001001010011000010101111011

Unit Addr: 0 1 2 3 4

• Integer (int): 32 bits (4 units)
Examples: 0, -1, 3434, 2999, ....

Numerical operations on integers return integers
3/2 = 1
Primitive Types

```
00011001001001010011000010101111011
```

Unit Addr: 0 1 2 3 4

- Boolean (boolean): True or False. 1 bit in theory.
  1 byte in C++. Up to 4 bytes in Java

- Boolean operators:
  
  ```
  && ---- and
  || ---- or
  ! ---- negation
  ```

- Write down the truth table if you are not sure
Primitive Types

Unit Addr: 0 1 2 3 4

- Character (char): 8 bits (256 characters. See ASCII table)

<table>
<thead>
<tr>
<th>Char</th>
<th>ASCII Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUL (null)</td>
<td>#32</td>
</tr>
<tr>
<td>start of heading</td>
<td>#33</td>
</tr>
<tr>
<td>start of text</td>
<td>#34</td>
</tr>
<tr>
<td>end of text</td>
<td>#35</td>
</tr>
<tr>
<td>end of transmission</td>
<td>#36</td>
</tr>
<tr>
<td>end undeniable</td>
<td>#37</td>
</tr>
<tr>
<td>acknowledge</td>
<td>#38</td>
</tr>
<tr>
<td>backspace</td>
<td>#39</td>
</tr>
<tr>
<td>vertical tab</td>
<td>#3A</td>
</tr>
<tr>
<td>NL line feed</td>
<td>#3B</td>
</tr>
<tr>
<td>carriage return</td>
<td>#3C</td>
</tr>
<tr>
<td>shift out</td>
<td>#3D</td>
</tr>
<tr>
<td>shift-in</td>
<td>#3E</td>
</tr>
<tr>
<td>data link escape</td>
<td>#3F</td>
</tr>
<tr>
<td>device control 1</td>
<td>#40</td>
</tr>
<tr>
<td>device control 2</td>
<td>#41</td>
</tr>
<tr>
<td>device control 3</td>
<td>#42</td>
</tr>
<tr>
<td>device control 4</td>
<td>#43</td>
</tr>
<tr>
<td>negative acknowledge</td>
<td>#44</td>
</tr>
<tr>
<td>synchronous idle</td>
<td>#45</td>
</tr>
<tr>
<td>extension of trans.</td>
<td>#46</td>
</tr>
<tr>
<td>cancel</td>
<td>#47</td>
</tr>
<tr>
<td>end of medium</td>
<td>#48</td>
</tr>
<tr>
<td>substitute</td>
<td>#49</td>
</tr>
<tr>
<td>escape</td>
<td>#50</td>
</tr>
<tr>
<td>file separator</td>
<td>#51</td>
</tr>
<tr>
<td>group separator</td>
<td>#52</td>
</tr>
<tr>
<td>unit separator</td>
<td>#53</td>
</tr>
</tbody>
</table>

- String is simply a sequence of characters + some methods to process the characters

Source: www.LookupTables.com
Primitive Types

[Binary code: 00011001001001010011000010101111011]

Unit Addr: 0  1  2  3  4

- Float (**float**): Numbers with fractional part. Lower precision & smaller range
  - 4 bytes
- Double (**double**): Higher precision & bigger range
  - 8 bytes

Examples: 0.454, 3.0, -1.2, 9.937654*10^20

What is the result of 1.0 + 0.0001?
What about 1.0 + 0.0000000000000000001?
Assignment compatibilities

- Usually, we need to put values of a certain type into variables of the same type.
- However, in some cases, the value will automatically be converted when types are different.

```
int age;
age = 10;
double length;
length = age;
```
Type Casting

- You can also cast a type into another type. But note the order of numerical types.

- $3 / 2 = 1$

- `(double) 3 / (double) 2 = 1.5$

- Try it your self:
  
  ```java
  System.out.println( 3/ 2);
  System.out.println( (float) 3/ (float) 2);
  ```

- What happens if you cast a double into int?
Order of Operations

• Arithmetic operations & parenthesis – check book if you are not sure

• Boolean: negation -> and -> or
• E.g.,  !true && (false || true) || true

• Mixed operations:
  – “The sum is” + 5 + 6
  – “The sum is” + (5 + 6)
Fun with Variables

• How do you swap the values of two variables?

• How do you swap the values of two integer variables?
Next Class

• Lab 0
• Bring your laptop & textbook
• To-do before the class:
  – Review our slides on creating objects & accessing objects’ methods