COMP 110-003
Introduction to Programming
Midterm Review

March 5, 2013

Haohan Li
TR 11:00 – 12:15, SN 011
Spring 2013
Announcements

• The grades for Lab 3 and Program 2 are released
  – Overall, most of you are doing quite well
  – Similar code has been detected. I will report officially next time
    • Some programs use the same techniques – please limit your discussions
Suggestions that I receive

• Your suggestions in Lab 4
  – Mostly mentioned: the instructor should show the procedure of writing a program, like the “case study” for loops
    • I will try to invent more “case studies” on class
    • Many new concepts can not be introduced using a full program. We have to continue the current way – using small pieces of code to introduce new concepts, and learning to combine them by assignments
Suggestions that I receive

• Your suggestions in Lab 4
  – More collaborations
    • That’s not going to happen!
    • I hope that you can use the lab time and office hour more effectively
    • Also you can send me emails. I usually respond quickly.

  – Another TA
    • That’s not going to happen, either...
    • Someone mentioned that the tutoring group was helpful
Suggestions that I receive

• Your suggestions in Lab 4
  – Focus on one topic, explain slowly and clearly
    • Next time when I ask “any questions”, please understand it as “do you want me to repeat”
    • Please don’t keep silent. I do reserve time for your questions but it was seldom used in recent lectures
  – Bring back daily jokes
Midterm Review

- Computer basics
- Primitive types and strings
- Branch statements and loop statements
- Classes and objects
Midterm Review

- **Computer basics**
- Primitive types and strings
- Branch statements and loop statements
- Classes and objects
Instructions

• An instruction is a sequence of 0’s and 1’s that represents a single operation on the computer
  – Example: 00000101 00000001 00000010
  – Means: ADD 1 2

    Instruction | Data

  – The output will be 3

• These 0’s and 1’s are called **bits**
  – Why only 0 and 1?
    • Because it is easy to make an electrical device that has only two stable states
Hardware vs Software (Abstractly)

• Software
  – An organized collection of instructions

• Hardware
  – Circuits that execute, store and interact with instructions
    • Execution: CPU
    • Storage: Memory
    • Interaction: Peripherals, like keyboards, monitors, networks
CPU (Central Processing Unit)

- It is the “brain” of the computer
  - CPU executes the instructions
  - CPU’s working routine
    - read instructions and data from memory
    - do calculation
    - write calculation results back to memory

- Intel Core i7 3.4 GHz
  - Executes *at most* 3,400,000,000 instructions per second
  - Recall: KB, MB, GB, TB
From Languages to Instructions

- The translator is called **compiler**
  - It is also a program
  - From human-readable to machine-readable

```java
class Hello {
    public static void main(String[] arguments) {
        // Program execution begins here
        System.out.println("Hello world.");
    }
}
```
Midterm Review

• Computer basics
• **Primitive types and strings**
• Branch statements and loop statements
• Classes and objects
Primitive Types

• Integer (byte, short, int, long)
  – 0, -3, 5, 43
• Floating-point number (float, double)
  – 0.5, 12.4863, -4.3
• Characters (char)
  – A, r, %, T
• Boolean (boolean)
  – true, false
Variable Declaration

• Syntax:
  – type variable_1, variable_2, ...;

• Examples:
  – int count, score, myInt;
  – char letter;
  – double totalCost, ratio;
How to name an identifier

• Naming rules:
  – Letters, digits(0-9)
  – First character *cannot* be a digit
  – No spaces
• Java is case sensitive
• Legal names
  – pinkFloyd, b3atles, eyeColor
• Illegal names
  – michael.bolton, kenny-G, 1CP
Assign and Change Variables

- `int changingVar = 0;`
  - Declare and assign value
  - `type variable = value;`

- `changingVar = 5;`
  - Assign/change value, variable must be declared before
  - `variable = value;`

- `changingVar = changingVar + 4;`
  - Can refer to itself
  - It means `newValue = oldValue + 4`. Now `changingVar = ?`
Special Assignment Operators

• Some operators are new to you
  – $total += 5; // is the same as$
  – $total = total + 5;$
  – $count++; // is the same as$
  – $count = count + 1;$

• They are created because
  – It’s shorter
  – Less possibility of making mistakes
Assignment Compatibilities

• You can only put small things into bigger things
  – byte->short->int->long->float->double

• Some examples
  – `myShort = myInt;` Wrong
  – `myByte = myLong;` Wrong
  – `myFloat = mybyte;` Right
  – `myLong = myInt;` Right

• Recall: `double d = int1 / int2;`
Type Casting

- You can ask the computer to change the type of values which are against the compatibility.
  - `myFloat = myDouble;
  - `myByte = myInt;
  - `myShort = myFloat;
  - `myFloat = (float)myDouble;
  - `myByte = (byte)myInt;
  - `myShort = (short)myFloat;

- You may lose information

- Recall: `int i = (int)(double1 / double2);`
Modular Arithmetic - %

- Remainder
  - $7 \% 3 = 1 \ (7 / 3 = 2, \text{ remainder } 1)$
  - $8 \% 3 = 2 \ (8 / 3 = 2, \text{ remainder } 2)$
  - $9 \% 3 = 0 \ (9 / 3 = 3, \text{ remainder } 0)$

- “clock arithmetic”
  - Minutes on a clock are mod 60

- Recall: what does $(a \% 2 == 0)$ mean?
indexOf() and substring()

String s = "2.5 + 3";
int p1 = s.indexOf(" ");
int p2 = s.lastIndexOf(" ");
System.out.println(p1 + "," + p2);
The output will be 3,5
indexOf() and substring()

<table>
<thead>
<tr>
<th>2</th>
<th>.</th>
<th>5</th>
<th>+</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

- String operand1 = s.substring(0, p1);
indexOf() and substring()

- String operand1 = s.substring(0, p1);
- String operator = s.substring(p1, p2);
indexOf() and substring()

- String operand1 = s.substring(0, p1);
- String operator = s.substring(p1, p2);
- String operand2 = s.substring(p2);
Midterm Review

- Computer basics
- Primitive types and strings
- **Branch statements and loop statements**
- Classes and objects
Boolean Expressions

• A combination of values and variables by comparison operators. Its value can only be **true** or **false**

• Example expressions
  – 5 == 3; // false
  – variable <= 6; // depending on the value of variable
    • What if variable is 5? What if variable is 6?
  – myInt != temp; // depending on both values
    • What if myInt is 0 and temp is 2? Am I lying?

• Syntax rule for if statement:
  – **if** *(boolean expression)*
    { statements; }

  **THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL**
## Logical Operators

**FIGURE 3.7** The Effect of the Boolean Operators `& &` (*and*), `||` (*or*), and `!` (*not*) on Boolean Values

<table>
<thead>
<tr>
<th>Value of $A$</th>
<th>Value of $B$</th>
<th>Value of $A &amp; &amp; B$</th>
<th>Value of $A \mid \mid B$</th>
<th>Value of $! (A)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>
More Complex Boolean Expressions

• Combination of && and ||
  – \( ((3 < 7) \lor (2 == 5)) \land ((4 \neq 2) \land (1 \leq 1)) \)
  – \( ((\text{true}) \lor (\text{false})) \land (\text{true}) \land (\text{true}) \)
  – (true) \land (true)
  – true

• if ( (I’m at Subway) \land (You’re at Subway) ) \lor \n  ( (I’m at Starbucks) \land (You’re at Starbucks) )
  
  { 
    I will meet you; 
  }

If and Else

- You can use only one if statement
  
  ```java
  if (boolean expression)
  { statements; }
  other statements;
  
  Other statements will always be executed
  ```

- You can also use an if-else statement
  
  ```java
  if (boolean expression)
  { statements 1; }
  else { statement 2; }
  
  If the expression is true, run statement 1, otherwise run statement 2
  ```
Multibranch and Switch Statement

```java
if (year == 1)
    System.out.println("freshman");
else if (year == 2)
    System.out.println("sophomore");
else if (year == 3)
    System.out.println("junior");
else if (year == 4)
    System.out.println("senior");
else if (year == 5)
    System.out.println("super senior");
else
    System.out.println("huh?");
```

```java
switch (year) {
    case 1:
        System.out.println("freshman");
        break;
    case 2:
        System.out.println("sophomore");
        break;
    case 3:
        System.out.println("junior");
        break;
    case 4:
        System.out.println("senior");
        break;
    case 5:
        System.out.println("super senior");
        break;
    default:
        System.out.println("unknown");
        break;
}
```
While Statement

• Flow of while statement
  – Start from expression evaluation
  – As long as it’s true, repeat instructions in brackets

```
while (count <= number) {
    System.out.println(count);
    count++;
}
```
Do-While Statement

- Flow of do-while statement
  - Start from body statements
  - Repeat instructions in brackets as long as the expression is true

```java
do {
    System.out.print(count);
count++;
} while (count <= number);
```
For Statement

- Flow chart
  - for (Initializing_Action; Boolean_Expression; Update_Action){
    Body;
  }

```java
for (count = 1; count <= number; count++) {
    // all the actions
}
```
Midterm Review

• Computer basics
• Primitive types and strings
• Branch statements and loop statements
• **Classes and objects**
Classes vs. Objects

• **Classes:**
  – What we can create
  – Specify the data to save

• **Objects:**
  – What have been created
  – Save actual data
Defining a class

```java
public class Student {
    public String name;
    public int classYear;
    public double GPA;
    public String major;
    // ...

    public String getMajor() {
        return major;
    }

    public void increaseYear() {
        classYear++;
    }
}
```
Methods

```java
public String getMajor()
{
    return major;
}
```

returns a String

```java
public void increaseYear()
{
    classYear++;
}
```

return type

returns nothing
public class Student
{
    public String name;
    public int classYear;
    // ...
    public void setName(String studentName)
    {
        name = studentName;
    }
    public void setClassYear(int year)
    {
        classYear = year;
    }
}
Local Variable Rule

- Usually, a variable is only accessible in its surrounding brackets

```java
public class Variable {
    String a = "a";

    public void f() {
        String b = "b";
        if (a.equals("b")) {
            String c = "c";
        }
    }
}
```
public/private Modifier

- **public**: there is no restriction on how you can use the method or instance variable
- **private**: can not directly use the method or instance variable’s name outside the class
public class Student
{
    public int classYear;
    private String major;
}

public class StudentTest{
    public static void main(String[] args){
        Student jack = new Student();
        jack.classYear = 1;
        jack.major = "Computer Science"; // ERROR!!!
    }
}

OK, classYear is public

Error!!! major is private
Sample Questions

- The Fibonacci sequence is defined such that a number in the sequence is the sum of the previous two numbers in the sequence such that the first two numbers are 0, 1.

- The Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, ...

- Write a function, fib that takes an integer as input and displays the Fibonacci sequence to the first value greater than the input.

- Example: fib(6) would display 0, 1, 1, 2, 3, 5, 8
Sample Questions

• What is the value of temp at the end of the loop for the following input? (23)
• 9 4 2 3 8 11 7

```java
System.out.print("Please input an int: ");
int input = kb.nextInt();
int temp = 0;
for(int i=0; i < 6; i++){
    if(input % 2 == 1){
        temp += input;
    }
    System.out.print("Please input an int: ");
    input = kb.nextInt();
}
```
Sample Questions

• Write a method that returns the *absolute value* of the integer passed in to the method. The *absolute value* of a number is its numerical value without regard to its sign. For example, the absolute value of -3 is 3. The absolute value of 3 is also 3. Fill in the return type for the method below, and then fill in the body.

```
public _____ absoluteValue(int num) {
}
```
public int absoluteValue(int num)
{
    if (num < 0)
        return -num;
    else
        return num;
}

Another possibility:
public int absoluteValue(int num)
{
    if (num < 0)
        num = -num;
    return num;
}