CS1101 Programming and Problem Solving

Dr. Gina Bai

Spring 2023

Logistics

- **PA01 – W, A, B, C** on **zyBook > Chap 11**
  - Due: Thursday, Jan 26, at 11:59pm
- **ZY-2B** on **zyBook > Assignments**
  - Due: Saturday, Jan 28, at 11:59pm

Recap

Q: What's the exact output of the following code?

```
int x = 7;
int y = 6;
x = y;
System.out.println("x is " + x);
```

x is 6

```
int x = -11;
x = -x;
System.out.println("x is " + x);
```

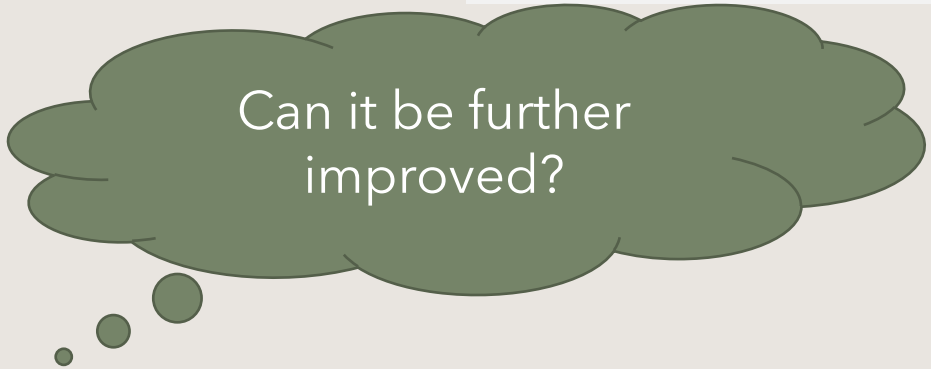
x is 11

```
double tax = 1.177777;
System.out.printf("%.2f", tax);
```

1.18

```java
public class ReceiptFormatted {
    public static void main(String[] args) {
        // Calculate total owed, assuming 7% tax and 18% tip.
        double subtotal = 38.0 + 40.0 + 30.0;
        double tax = subtotal * 0.07;
        double tip = subtotal * 0.18;
        double total = subtotal + tax + tip;

        System.out.printf("%-12s $%7.2f%n", "Subtotal", subtotal);
        System.out.printf("%-12s $%7.2f%n", "Tax", tax);
        System.out.printf("%-12s $%7.2f%n", "Tip", tip);
        System.out.printf("%-12s $%7.2f%n", "Total", total);
    }
}
```



Constants

zyBook Chap 2.16

What?

A constant is a **fixed value**

- The value of a constant can be **set only at declaration**; it **cannot be reassigned.**

Why?

- Constants help to reduce complexity by eliminating magic numbers.
  - Magic number: numbers that make the program work, but have no obvious meaning in the program.
- Constants make our programs more readable and adaptable.

Class Constants

Declare and initialize **within the class** but **outside of the method**

public static final <type> <CONSTANT_NAME> = <value>;

- **Visible to the whole class**
- Naming convention: **All uppercase** with words separated by **underscores**

```
public class ClassConstantDemo {

    /** Class constant for interest rate */
    public static final double INTEREST_RATE = 3.5;

    public static void main(String[] args) {
        // More code here...
    }
}
```

Javadoc the class constants

Constants within a Method

Declare and initialize **within the method**

final <type> <CONSTANT_NAME> = <value>;

- **Visible within the method after it is declared**
- Naming convention: **All uppercase** with words separated by **underscores**

```
public class ClassConstantDemo {
    public static void main(String[] args) {

        /** Constant for interest rate */
        final double INTEREST_RATE = 3.5;

        // More code here...
    }
}
```

```java
import java.util.Scanner;

public class ReceiptInteractive {

    /** Class constant for tax rate, 7% */
    public static final double TAX_RATE = 0.07;
    /** Class constant for tip rate, 18%*/
    public static final double TIP_RATE = 0.18;

    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        // Ask the user to enter the subtotal
        System.out.print("Enter the subtotal: ");

        // Read in the user input and store into subtotal
        double subtotal = input.nextDouble();

        double tax = subtotal * TAX_RATE;
        double tip = subtotal * TIP_RATE;
        double total = subtotal + tax + tip;

        System.out.printf("%-12s $%7.2f%n", "Subtotal", subtotal);
        System.out.printf("%-12s $%7.2f%n", "Tax", tax);
        System.out.printf("%-12s $%7.2f%n", "Tip", tip);
        System.out.printf("%-13s$%7.2f%n", "Total", total);
    }
}
```

```
$ javac ReceiptInteractive.java
$ java ReceiptInteractive
Enter the subtotal: 108.0
Subtotal     $ 108.00
Tax          $   7.56
Tip          $  19.44
Total        $ 135.00
```

Math Class

zyBook Chap 2.17

Math Class

- A part of the Java Class Library – `java.lang`
  - Default package
  - No need to import the Math Class explicitly

- Contains predefined constants and common mathematical methods
  - The methods generate/**return** values

How to Use Math Class

Since the mathematical methods and constants are in another class, we use ***dot notation*** to call them:

- **<ClassName>.<methodName>(<parameter(s)>)**
  - `Math.sqrt(4); // square root of 4`

- **<ClassName>.<CONSTATNT_NAME>**
  - `Math.E   // Euler's number`
  - `Math.PI  // π`

Common Math Methods

`sqrt(double x)`

```
Math.sqrt(4.0)  // 2.0
```

- the square root of x in double

`pow(double x, double y)`

```
Math.pow(3.0, 2.0)  // 9.0
```

- the value of x raised to the power of y in double

`abs(int x)`

`abs(double x)`

```
Math.abs(-4)  // 4
```

- the absolute value of x

Common Math Methods

`min(int x, int y)`

`min(double x, double y)`

- the smaller of x and y

```
Math.min(-4, 5)  // -4
```

`max(int x, int y)`

`max(double x, double y)`

- the greater of x and y

```
Math.max(-4, 5)  // 5
```

Common Math Methods

`ceil(double x)`

- the smallest whole number (in double) that is greater than or equal to x

`floor(double x)`

- the largest whole number (in double) that is less than or equal to x

`round(double x)`

- the closest long to x

```
Math.ceil(3.5);  // 4.0

Math.floor(3.5);  // 3.0

Math.round(3.5);  // 4
```

Rounding Real Numbers to N decimal places

Three steps to round a real number to N decimal places:

1. Multiply by 10^N
2. Round
3. Divide by 10^N

```
double result = 1.0 / 3.0;       // 0.333333333333

result = result * 100;           // 33.333333333

result = Math.round(result);     // 33.0

result = result / 100;           // 0.33
```

Strings

zyBook Chap 1.5, 2.9, 4.14, 4.15, 4.16

String class

- The `String` class represents character strings.
- `String` objects are immutable and cannot change.
- `String` objects can be created and assigned like primitive values:
  - **`String <name> = "<text>";`**
  - **`String <name> = <expression>;`**

- For example:

```
String department = "CS";
int courseNum = 1101;
String course = department + courseNum;  // "CS1101"
```

String Indexing

- String objects consist of a list of characters (char)
- Characters are numbered internally with an index
- **Index starts at 0**
- For example,

```
String school = "Vandy";
```

0	1	2	3	4
V	a	n	d	y

How to determine if two String objects are equal?

public boolean equals(Object anObject)

- Parameter: anObject – The object to compare this String against
- Returns: **true** if the given object represents the same sequence of characters as this String, **false** otherwise
- For example,

```
String school = "Vandy";
school.equals("Vandy");       // true
school.equals("Vanderbilt");  // false
```

How many characters are in the String?

`public int length()`

- Returns: the length of the sequence of characters represented by this object.
- For example,

```
String school = "Vandy";
int numLetter = school.length();  // 5
```

0	1	2	3	4
V	a	n	d	y

What is the N-th character of the String?

`public char charAt(int index)`

- Parameter: index – the index of the char value.
- Returns: the char value at the specified index of this string.
- Throws: IndexOutOfBoundsException if the index argument is negative or not less than the length of this string.
- For example,

```
String school = "Vandy";
school.charAt(0);  // 'V'
school.charAt( school.length() – 1 );  // 'y'
school.charAt( school.length() );      // IndexOutOfBoundsException
```

0	1	2	3	4
V	a	n	d	y

Where is a substring in the String?

`public int indexOf(String str)`

- Parameter: str – the substring to search for.
- Returns: the index of the first occurrence of the specified substring, or -1 if there is no such occurrence.
- For example,

```
String school = "Vandy";
school.indexOf("n");    // 2
school.indexOf("an");   // 1
school.indexOf("ad");   // -1
```

0	1	2	3	4
V	a	n	d	y

Substrings

`public String substring(int beginIndex)`

- Parameter: **beginIndex** – the beginning index, **inclusive**.
- Returns: the specified substring.
- Throws: IndexOutOfBoundsException if beginIndex is negative or larger than the length of this String object.
- For example,

```
String school = "Vandy";
school.substring(1);  // "andy"
```

0	1	2	3	4
V	a	n	d	y

`public String substring(int beginIndex, int endIndex)`

- Parameters:
  beginIndex – the beginning index, **inclusive**.
  endIndex – the ending index, **exclusive**.
- Returns: the specified substring.
- Throws: IndexOutOfBoundsException
  - the beginIndex is negative, or
  - the endIndex is larger than the length of this String object, or
  - the beginIndex is larger than endIndex.
- For example,

```
String school = "Vandy";
school.substring(0, 2);  // "Va"
```

0	1	2	3	4
V	a	n	d	y

Uppercase and Lowercase

`public String toLowerCase()`

- Returns: the String, converted to lowercase.

`public String toUpperCase()`

- Returns: the String, converted to uppercase.

- For example,

```
String school = "Vandy";
String lower = school.toLowerCase();   // "vandy"
String upper = school.toUpperCase();   // "VANDY"
```

How to compare two String objects?

`public int compareTo(String anotherString)`

- Parameter: anotherString – the String to be compared.
- Returns:
  - the value 0 if the argument string is equal to this string;
  - a value less than 0 if this string is lexicographically less than the string argument;
  - a value greater than 0 if this string is lexicographically greater than the string argument.
- For example,

```
String school = "Vandy";
school.compareTo("Vandy");  // 0
school.compareTo("VANDY");  // 32
school.compareTo("vandy");  // -32
```

Q: Find the exact output of the following code

```
public class StringExample {

    public static void main(String[] args) {
        String question = "How are you?";
        String response = "I am fine. Thanks.";

        System.out.println(question.length());   // 12
        System.out.println(response.length());   // 18
        System.out.println(question.length() + response.length()); // 30

        String sub1 = question.substring(3, 7);
        System.out.println(sub1.toUpperCase());  //  ARE

        String sub2 = response.substring(7);
        System.out.println(sub2.toLowerCase());  // ne. thanks.
    }
}
```

Reading String Token(s) from Console

- **`public String next()`**
  - Reads and returns user input as a String

- **`public String nextLine()`**
  - Reads and returns and entire line of user input as a String