

Object Oriented Paradigms

College Requirements

CSCR2105



Classes

Lecture 2.2

The String Class

- Because strings are so common, we don't have to use the **new** operator to create a **String** object

```
title = "Java Software Solutions";
```

- This is special syntax that works only for strings objects
- Each string (enclosed in double quotes) represents a **String** object

String Methods

- Once a **String** object has been created, neither its value nor its length can be changed
- Thus we say that **an object** of the **String** class is *immutable*
- However, several methods of the **String** class return new **String** objects that are modified versions of the original
- See the list of **String** methods on **chapter 16**
- You can also use the API documentation

String Indexes

- o It is occasionally helpful to refer to a particular character within a string
- o This can be done by specifying the character's numeric *index*
- o The indexes begin at zero in each string
- o In the string "Hello", the character 'H' is at index 0 and the 'o' is at index 4

Example

```
String phrase = new String ("Change is inevitable");
String m1, m2, m3, m4;
System.out.println ("Original string: \"\" + phrase + "\"");
System.out.println ("Length of string: " + phrase.length());
m1= phrase.concat (" , except from vending machines.");
m2= m1.toUpperCase();
m3= m2.replace ('e', 'x');
m4= m3.substring (3, 30);
// Print each mutated string
System.out.println ("Text #1: " + m1);
System.out.println (" Text #2: " + m2);
System.out.println (" Text #3: " + m3);
System.out.println (" Text #4: " + m4);
System.out.println (" Text length: " + m4.length());
```

The Math Class

- o The `Math` class is part of the **`java.lang`** package
- o The `Math` class contains methods that perform various mathematical functions
- o These include:
 - o absolute value
 - o square root
 - o exponentiation
 - o trigonometric functions

The Math Class

- o The methods of the **Math** class are **static** *methods* (also called *class methods*)
- o Static methods can be invoked through the class name – no object of the Math class is needed

```
value = Math.cos(90) + Math.sqrt(delta);
```

- o See Chapter 6


```
int a, b, c; // ax^2 + bx + c
double dis, root1, root2;
Scanner scan = new Scanner (System.in);
System.out.print ("Enter the coefficient of x squared: ");
a = scan.nextInt();
System.out.print ("Enter the coefficient of x: ");
b = scan.nextInt();
System.out.print ("Enter the constant: ");
c = scan.nextInt();
dis= Math.pow(b, 2) - (4 * a * c);
root1 = ((-1 * b) + Math.sqrt(dis)) / (2 * a);
root2 = ((-1 * b) - Math.sqrt(dis)) / (2 * a);
```

+

The Date Class

Java provides a system-independent encapsulation of date and time in the java.util.Date class. You can use the Date class to create an instance for the **current date and time** and use its toString method to return the date and time as a string.

The + sign indicates
public modifier

java.util.Date	
+Date()	
+Date(elapseTime: long)	
+toString(): String	
+getTime(): long	
+setTime(elapseTime: long): void	

Constructs a Date object for the current time.

Constructs a Date object for a given time in milliseconds elapsed since January 1, 1970, GMT.

Returns a string representing the date and time.

Returns the number of milliseconds since January 1, 1970, GMT.

Sets a new elapse time in the object.

The Date Class Example

For example, the following code

```
java.util.Date date = new java.util.Date();  
System.out.println(date.toString());
```

displays a string like Mon Nov 14 09:50:19
EST 2016.

The Random Class

You have used `Math.random()` to obtain a random double value **between 0.0 and 1.0** (excluding 1.0). A more useful random number generator is provided in

java.util.Random

+Random()	Constructs a Random object with the current time as its seed.
+Random(seed: long)	Constructs a Random object with a specified seed.
+nextInt(): int	Returns a random int value.
+nextInt(n: int): int	Returns a random int value between 0 and n (exclusive).
+nextLong(): long	Returns a random long value.
+nextDouble(): double	Returns a random double value between 0.0 and 1.0 (exclusive).
+nextFloat(): float	Returns a random float value between 0.0F and 1.0F (exclusive).
+nextBoolean(): boolean	Returns a random boolean value.

The Random Class Example

If two Random objects have the same seed, they will generate identical sequences of numbers. For example, the following code creates two Random objects with the same seed 3.

```
Random random1 = new Random(3);  
System.out.print("From random1: ");  
for (int i = 0; i < 10; i++)  
    System.out.print(random1.nextInt(1000)+" ");
```

```
Random random2 = new Random(3);  
System.out.print("\nFrom random2: ");  
for (int i = 0; i < 10; i++)  
    System.out.print(random2.nextInt(1000)+" ");
```

From random1: 734 660 210 581 128 202 549 564 459 961

From random2: 734 660 210 581 128 202 549 564 459 961

Class Libraries

- o A class **library** is a collection of classes that we can use when developing programs
- o The **Java standard class library** is part of any Java development environment
- o Its classes are not part of the Java language per se, but we rely on them heavily
- o Various classes we've already used (**System**, **Scanner**, **String**) are part of the Java standard class library
- o Other class libraries can be obtained through third party vendors, or you can create them yourself

Packages

- o The classes of the Java standard class library are organized into *packages*
- o Some of the packages in the standard class library are:

Package

Purpose

java.lang

General support

java.applet

Creating applets for the web

java.awt

Graphics and graphical user interfaces

javax.swing

Additional graphics capabilities

java.net

Network communication

java.util

Utilities

javax.xml.parsers

XML document processing

The import Declaration

- When you want to use a class from a package, you could use its *fully qualified name*

```
java.util.Scanner;  
java.util.Random;
```

- Or you can *import* the class, and then use just the class name

```
import java.util.Scanner;
```

- To import all classes in a particular package, you can use the ***** wildcard character

```
import java.util.*;
```

The import Declaration

- o All classes of the **java.lang** package are imported automatically into all programs
- o It's as if all programs contain the following line:

```
import java.lang.*;
```
- o That's why we didn't have to import the **System** or **String** classes explicitly in earlier programs
- o The **Scanner** class, on the other hand, is part of the **java.util** package, and therefore must be imported

Summary

- o String
- o Math
- o Date
- o Random



NOW:

Waiting for your questions and comments

**Object-Oriented
Programming: Inheritance**