Introduction to java

COMP1030 Lecture #3

Housekeeping

- Our goal - 100% pass rate in this course.
- Review the lecture slides ahead of time.
- Review the lecture slides after class with a study group.
- Repeat the lab at home 1-2 times.
- Take notes during class.
- Weekly optional tutorials will begin next week (Jan 27)- watch for blackboard announcement

Review

- Comments
- Semicolon, blocks, whitespace
- Identifiers
- Java keywords
- Types (boolean, textual, integral, floating)
- Arithmetic operators
- Equality and relational operators
- Programming conventions
- Import Statements

Instance variables, setters, getters

- An instance variable is specific to the object that is created (instantiated) from a class.
- Instance variables maintain data for each object (instance) of the class.
- Instance variables are <u>not</u> shared among instances.

Instance variables, setters, getters

```
public class Account
{
        private String name;
        private int accountNumber;
}
```

Instance variables, setters, getters

- private variables or methods are accessible only to methods of the class in which they are declared.
- public variables/methods are available outside the class
- Instance variables are typically listed first in the classes body, and are typically private to enforce security through data hiding (encapsulation).

Instance variables, setters, getters

- Every instance variable has a default initial value if you do not specify an initial value.
- Each object (instance of a class) has its own copy of the class's instance variables.
- A class normally contains methods (setters, getters) to manipulate or access its instance variables

Method signature

Parameter

```
public void setName(String usersName)
{
        name = usersName;
}
```

- Variables declared in the method signature are called parameters, and are local variables which can only be used in that method. When a method terminates the values of its local variables are lost.

Method signature

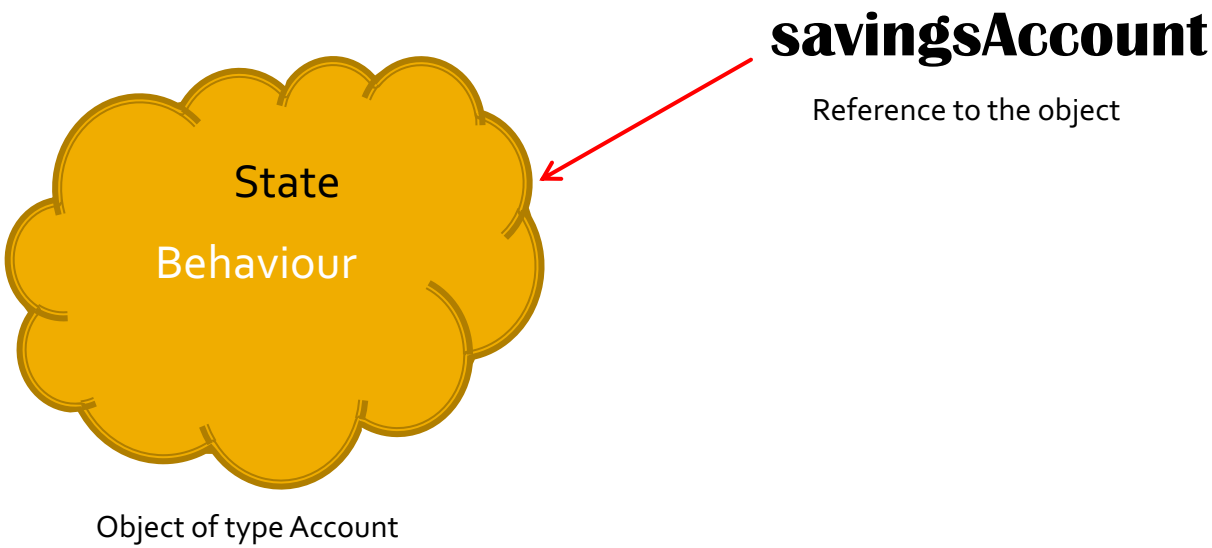
```
public String getName()
{
        return name;
}
```

- Methods can return a value to the caller by using the keyword 'return'.
- If a method does not return anything its return type is 'void'.

Instantiating a new object

- The keyword "new" creates a new object of the specified class.

Account savingsAccount = new Account();



Constructors

- A constructor is a block of code (method) that is called when a new object is created using the new keyword.
- A constructor does not have a return type.
- The name of the constructor **must have** the same name as the class.
- A constructor can take a comma separated parameter list to provide initial values for instance variables.

Constructors

```
public Customer (String first, String last)
{
        firstName = first;
        lastName = last;
}

Customer c = new Customer("Bill", "Black");
```

Constructors

- A class can have multiple constructors (constructor overloading).
- Each constructor must have a different signature. (# of parameters, type of parameters, or order of parameters)
- Constructors are not inherited.
- If you do not write a constructor the java compiler will create a default constructor with no parameters.

Constructors

- If you declare any constructors, java will <u>NOT</u> create the default constructor.

```
Customer c = new Customer();
```

Understanding the main method

public static void main (String [] args)

- The main method is the entry point to your program as it is called automatically by the JVM. Most other methods must be explicitly called.

Understanding the main method

public static void main (String [] args)

- static – method can be called <u>without</u> having to instantiate an object.
- public – can be called by any object.
- void – no return type
- String [] args – accepts a single argument of an array ([]) of Strings.

Understanding the main method

public static void main (String [] args)

- The array ([]) of Strings is a mechanism through which the runtime system passes information (command line arguments) to your application.
- This allows the user to use command line arguments to affect the operation of the application without re-compiling it.

Review of the String class

- A String is an Object.
- A String is a sequence of characters.
- A String is immutable – once created it cannot be changed.

Understanding the String class

- Strings can be created in two ways:

```
String s1 = "Hello";
String s2 = "Hello";
```

- Note no use of the "new" keyword. (implicit instantiation)
- In this case java creates only ONE String object with two references pointing to it.

Understanding the String class

```
String s1 = new String("Hello");
String s2 = new String("Hello");
```

- In this case java creates two objects.

Argument Promoting and Casting

- The Math.sqrt() method expects a double argument.
- However Math.sqrt(4); is valid because java will automatically promote (convert) the int 4 to a double because no data will be lost in this promotion.

Argument Promoting and Casting

Promotions allowed for primitive types

Type	Valid promotions
double	None
float	double
long	float or double
int	long, float or double
char	int, long, float or double
short	int, long, float or double
byte	short, int, long, float or double
boolean	None (boolean values are not considered to be numbers in Java)

Argument Promoting and Casting

- Converting a double to an int will truncate the fractional portion of the double value, therefore part of the value is lost.

- If one attempts to use a type that requires a demotion (converting to a lower type) the compiler will error out.

- To force conversion (essentially saying to the compiler, I know this conversion might cause loss of information, but in this situation it is ok) we can use the cast operator.

Argument Promoting and Casting

Example

- A method called calculateResult requires an argument of type int.

- To pass a double we would need to cast the double to an int: (this is known as type casting)

```
ref.calculateResult((int)32.64);
```

Code Example

```java
public class StudentRecord
{
    private String fullName;
    private int studentNumber;

    public StudentRecord (String fullName, int inStudentNumber)
    {
        this.fullName = fullName;
        studentNumber = inStudentNumber;
    }

    public void setFullName(String fullName)
    {
        this.fullName = fullName;
    }

    public String getFullName()
    {
        return fullName;
    }
}

public class StudentRecordTestHarness
{

    public static void main(String []args)
    {
        StudentRecord sr1 = new StudentRecord("Philip Smith", 12345);
        System.out.println(sr1.getFullName());
        sr1.setFullName("Philip Smythe");
        System.out.println(sr1.getFullName());
    }
}
```

```
$javac StudentRecordTestHarness.java
$java -Xmx128M -Xms16M StudentRecordTestHarness
Philip Smith
Philip Smythe
```

Output

Java Documentation

https://docs.oracle.com/javase/8/docs/api/

Java™ Platform Standard Ed. 8

All Classes All Profiles

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event
java.awt.font

All Classes

AbstractAction
AbstractAnnotationValueVisitor6
AbstractAnnotationValueVisitor7
AbstractAnnotationValueVisitor8
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractChronology
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeCon
AbstractDocument.Content
AbstractDocument.ElementEdit
AbstractElementVisitor6
AbstractElementVisitor7
AbstractElementVisitor8
AbstractExecutorService
AbstractInterruptibleChannel
AbstractLayoutCache
AbstractLayoutCache.NodeDime
AbstractList
AbstractListModel
AbstractMap
AbstractMap.SimpleEntry
AbstractMap.SimpleImmutableE
AbstractMarshallerImpl
AbstractMethodError
AbstractOwnableSynchronizer
AbstractPreferences
AbstractProcessor

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

Java™ Platform Standard Ed. 8

PREV NEXT FRAMES NO FRAMES

Java™ Platform, Standard Edition 8 API Specification

This document is the API specification for the Java™ Platform, Standard Edition.

See: Description

Profiles

- compact1
- compact2
- compact3

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within applications.
java.awt.dnd	Drag and Drop is a direct manipulation gesture found in many Graphical User Interface systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.
java.awt.event	Provides interfaces and classes for dealing with different types of events fired by AWT components.
java.awt.font	Provides classes and interface relating to fonts.