COMP 110/401
OBJECTS

Instructor: Prasun Dewan
Prerequisites

- Theater (optional)
Script components are abstract.

So are program components!
CONVENTIONAL LANGUAGES

Program

Procedures

Declarations

Statements

If Statement

While Statement

Script analogy works for conventional programming

For O-O programming analogy is physical objects
PROGRAM COMPONENTS ~ PHYSICAL OBJECTS

Natural Objects

Manufactured Objects

~ Program Components
Invoking new operation on class ~ ordering new car from factory
CLASSIFICATION THROUGH FACTORIES
CLASSIFICATION THROUGH CLASSES
A Simple Class: ASquareCalculator

```java
public class ASquareCalculator {
    public int square(int x) {
        return x*x;
    }
}
```

No static because operation on instance of ASquareCalculator rather than on the class ASquareCalculator

Methods, parameters and return value?
**Nature of a Function**

<table>
<thead>
<tr>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter Values</td>
<td>Result Values</td>
</tr>
<tr>
<td>1, 2, 3, ...</td>
<td>1, 4, 9, ...</td>
</tr>
</tbody>
</table>

**Mapping**
Mathematics vs. Java Function Syntax

**Math**

square: $\mathbb{I} \to \mathbb{I}$

\[
square(x) = x^2
\]

**Java**

```java
public int square(int x)
{
    return x*x;
}
```

Mathematical functions may not be transferred to the computer; and if they are, they are stored as unstructured text.
PROGRAMMING ENVIRONMENT

Source Code Editor

Mechanisms a way to run compiler and interpreter

Mechanisms to determine the location of libraries

Error reporting

Console for input and output
EXAMPLE PROGRAMMING Environments

- Complexity of simple tasks
  - Bare Bones: text editor, command window
  - Dr. Java
  - BlueJ
  - Eclipse

- Complexity of difficult tasks

Free to use any programming environment

We will provide help with Eclipse
public class ASquareCalculator {

    public int square(int x) {
        return x*x;
    }

}
Class C is saved in a file called C.Java

Programming environment may automatically create file name from class name.
Java Translation/Execution Process

1. **Source Code**
2. **Compiler**
   - javac
3. **Platform Independent Intermediate Code**
4. **Interpreter**
   - java
5. **Platform Specific Machine Code**
6. **Processor**
COMPILING A CLASS

D:\dewan_backup\Java\JavaTeaching\src>javac ASquareCalculator.java

ASquareCalculator.class

3/21/2012 11:04 AM  CLASS File
**CLASS FILE**

Object rather than source code

Object (bin) and source (src) usually kept in separate directories (folders)
public class ASquareCalculator
{
    public int square(int x) {
        return x*x;
    }
}
**Ordering and Using a Manufactured Object**

- **Factory**: Manufactured by
- **Operations**: Accelerate, Brake
- **More control**
- **More convenient and necessary for beginners and non-drivers**
- **You can do it yourself**
- **You can give orders to your chauffeur**
**Manual Driving: Writing our own Main**

```java
public class ASquareCalculator {
    public int square(int x) {
        return x*x;
    }
}
```

```java
public class SquareCalculatorDriver {
    public static void main(String[] args) {
        System.out.println((new ASquareCalculator()).square(5));
    }
}
```

We can now “run” main to” start operations on the factory”
ANALOGY FOR INSTANCE METHODS

Factory manufactured by

Operations
- accelerate
- brake

Class

Program Object

Methods
- execute
- invoke
- call
- add
- subtract

Program Object

Invoking new operation on class ~ ordering new car from factory
ANALOGY FOR STATIC (CLASS) METHODS

Static methods are invoked on a class rather than an class instance.

Operations
- accelerate
- brake

Factory
- manufactured by
- start operations

Class
- main

Program Object
- execute
- invoke
- call

Methods
- add
- subtract
CALLING MAIN FROM OUR PROGRAM

```java
public class SquareCalculatorDriver {
    public static void main (String[] args) {
        System.out.println (new ASquareCalculator().square());
    }
}

public class SquareCalculatorDriver Driver{
    public static void main (String[] args) {
        String myArgs[] = {};
        SquareCalculatorDriver .main(myArgs);
    }
```
public class SquareCalculatorDriver {
    public static void main (String[] args) {
        System.out.println ((new ASquareCalculator()).square
            (Integer.parseInt(args[0])));
    }
}
int product = 1;
int nextNum = Console.readInt();
while (nextNum >= 0) {
    product = product* nextNum;
    nextNum = Console.readInt();
}
System.out.print (product);
public class Console {
    static BufferedReader inputStream =
        new BufferedReader(new InputStreamReader(System.in));
    public static int readInt() {
        try {
            return Integer.parseInt(inputStream.readLine());
        } catch (Exception e) {
            System.out.println(e);
            return 0;
        }
    }
    public static String readString() {
        try {
            return inputStream.readLine();
        } catch (Exception e) {
            System.out.println(e);
            return "";
        }
    }
    ... //other methods
}
REAL LIFE ANALOGY

Warehouse, dealership
public class SquareCalculatorDriver
{
    public static void main (String[] args)
    {
        System.out.println ( (new ASquareCalculator()).square(5) );
    }
}
JAVA TRANSLATION PROCESS

Source Code

Compiler

Platform
dependent
Intermediate Code

Interpreter

Platform
specific
Machine Code

Processor

javac

java
Main Method Details

Main method has predefined header.

```java
public static void main (String[] args) {
    ....
}
```

All methods must be in some "class"

```java
public class SquareCalculatorDriver {
    public static void main (String[] args) {
        System.out.println ((new ASquareCalculator()).square(5));
    }
}
```

The Java interpreter calls main and provides its user-specified arguments. Public means interpreter can access main.
public class SquareCalculatorDriver
{
    public static void main (String[] args)
    {
        System.out.println (
            (new ASquareCalculator()).square(5));
    }
}

public class SquareCalculatorDriver
{
    public static void main (String[] args)
    {
        System.out.println ( 
            (new ASquareCalculator()).square
                (Integer.parseInt(args[0])));
    }
}
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}

public class SquareCalculatorDriver
{
    public static void main (String[] args)
    {
        System.out.println (  
            (new ASquareCalculator()).square(5));
    }
}
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}

public class SquareCalculatorDriver
{
    public static void main (String[] args)
    {
        System.out.println ( (new ASquareCalculator()).square(341));
    }
}

Calculates 341*341

For different parameter values must modify, compile and run program
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}

public class SquareCalculatorDriver
{
    public static void main (String[] args)
    {
        System.out.println(
            (new ASquareCalculator()).square
            (Integer.parseInt(args[0])));
    }
}

101 students must understand arrays
Must run program multiple times
Can get value from user as a main arg
User Input

Must run program multiple times

Unless we write tedious loops with prompting

Can get value from user

Can we build an interactive tool to have our cake and eat it too

public class SquareCalculatorDriver
{
    public static void main (String[] args)
    {
        System.out.println (new ASquareCalculator(Console.readInt()));
    }
}

public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
SELF DRIVEN PROGRAM DEVELOPMENT

Editor

ASquareCalculator Source Code

Compiler

ASquareCalculator Object (byte) Code

ASquareCalculator Instance

SquareCalculatorDriver

Interpreter

creates

reads

writes

instantiates

calls

main

calls

square
CHAUFFEUR-DRIVEN PROGRAM DEVELOPMENT

Editor

ASquareCalculator Source Code

Compiler

ASquareCalculator Object (byte) Code

ObjectEditor

Interpreter

main

calls

calls

instantiates

writes

reads

calls

square
Running ObjectEditor Main

java –classpath .;oeall20.jar bus.uigen.ObjectEditor
RUNNING OBJECT EDITOR MAIN
RUNNING OBJECT EDITOR MAIN

- src
- JRE System Library [jre1.6.0_04]
- Referenced Libraries
  - oeall20.jar - D:\dewan_backup\Java\lib
  - bus.ui/gen
    - AClassDescriptorListener.class
    - AddComponentEvent.class
    - ADynamicEnumeration.class
    - AListenableHashtableAttributeRegisterer_1.class
    - AListenableHashtableAttributeRegisterer.class
    - AListenableVectorTesterDriver.class
  - myLockManager.class
  - NodeData.class
  - NodeData.class
  - ObjectEditor.class
  - ObjectEditorApplet.class
  - ObjectEditorAR.class
INSTANTIATING ASQUARECALCULATOR
ASquareCalculator Instance
INVOKING A METHOD AND VIEWING THE RESULT
Invoking operations is essentially form filling
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}

public class SquareCalculatorDriver
{
    public static void main (String[] args)
    {
        System.out.println (
            (new ASquareCalculator()).square(341)
        );
    }
}
INVOKING A METHOD AND VIEWING THE RESULT
OE Capabilities

Interactive Instantiation (for beginners)

Interactive method invocation (for convenience)

Typically want to create instance in our main but invoke methods interactively
public class SquareCalculatorEditor {
    public static void main(String[] args) {
        bus.uigen.ObjectEditor.edit(new ABMICalculator());
    }
}

We order the car but chauffeur drives

ObjectEditor main?
OBJECTEDITOR vs. ASQUARECALCULATOR

Two objects with different methods

Object editor used to edit itself

Object editor used to edit ASquareCalculator
Meta Circularity: Object Editor Main

```java
public class ObjectEditor {
    public static void main (String[] args) {
        bus.uigen.ObjectEditor.edit(new ObjectEditor());
    }
}
```

Meta circularity/Bootstrapping; An service provider provides itself the service

- Compiler compiles itself
- OS loads and runs itself
- ObjectEditor edits itself
**Doing Our Own Instantiation**

```java
public class ObjectEditor {
    public static void main (String[] args) {
        bus.uigen.ObjectEditor.edit(new ObjectEditor());
    }
}
```

Can instantiate and edit objects without using knowing about or using main
THREE APPROACHES

public class SquareCalculatorEditor {
    public static void main(String[] args) {
        bus.uigen.ObjectEditor.edit(new ABMICalculator());
    }
}

public class SquareCalculatorDriver {
    public static void main (String[] args)
    {
        System.out.println ( (new ASquareCalculator()).square(5));
    }
}
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}

public class SquareCalculatorDriver
{
    public static void main (String[] args)
    {
        System.out.println (
            (new ASquareCalculator()).square(5));
    }
}
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}

public class SquareCalculatorUser
{
    public static void main (String[] args)
    {
        ASquareCalculator squareCalculator = new ASquareCalculator();
        System.out.println(squareCalculator.square(5));
        System.out.println(squareCalculator.square(341));
    }
}
Instance of

Class & Program Object

Factory manufactures

Operations

accelerate, brake

Methods

execute, invoke, call

add, subtract
**JAVA INSTANCEOF BOOLEAN OPERATOR**

(new ASquareCalculator()) instanceof ASquareCalculator

true

((new ASquareCalculator()) instanceof Integer)

false

((new ASquareCalculator()) instanceof ObjectEditor)

false

true and false are values of Java type boolean

instanceof is Java keyword

If class of object o is T then o instanceof T is true
ANOTHER SIMPLE CLASS: ABMICALCULATOR

**Specification:**
Given an integer x, calculate the square of x.

```java
public class ASquareCalculator {
    public int square(int x) {
        return x*x;
    }
}
```

**Specification:**
Given the weight (kg) and height (m) of a person, calculate the person’s body mass index a.k.a. BMI.

```java
?```
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}

public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}

int weight, int height

ABMICalculator

public class ABMICalculator
{
    public int square(int x)
    {
        return x*x;
    }
}

return weight/(height*height)
public class ABMICalculator {
    public int calculateBMI(int weight, int height) {
        return weight/(height*height);
    }
}
public class ABMICalculator
{
    public double calculateBMI(double weight, double height)
    {
        return weight/(height*height);
    }
}

ABMICalculator
```java
public class ABMICalculator {
    public double calculateBMI(double weight, double height) {
        return weight / (height * height);
    }
}
```
**FORMAL VS. ACTUAL PARAMETERS**

```java
public class ABMICalculator {
    public double calculateBMI(double weight, double height) {
        return weight/(height*height);
    }
}
```

Invoke `calculateBMI` with actual parameters, resulting in memory usage:

- **Weight**: 74.98
- **Height**: 1.94
JAVA CASE CONVENTIONS

Start Class Names With Upper Case Letters

aBMICalculator  ABMICalculator

Start Variable and Method Names With Lower Case Letters

weight  Weight
Square  square

Start Variable and Method Names With Lower Case Letters
Each New Word in the Name Starts with a Capital Letter

converttoinches  convertToInches
ABMICalculator  aBMICalculator
**OBJECTEDITOR CHANGES CASE**

Method name starts with capital letter

Adds spaces to names

Different conventions for programmers and end users.
**LEGAL CLASS?**

```
public class Class
{
    public int square(int x)
    {
        return x*x;
    }
}
```
Identifiers

Reserved Words (Keywords)
- int, double, class, return, public

Program-defined Names
- Variable names, method names, class names
  - Must begin with a letter
  - Other characters can be letters, digits, or underscore "_"
  - calculate_bmi2
  - calculateBmi3
**Identifier Name Choices**

```java
public class ABMICalculator {
    public double calculateBMI(double weight, double height) {
        return weight/(height*height);
    }
}

public class C {
    public double f(double p1, double p2) {
        return p1/(p2*p2);
    }
}
```

Use mnemonic identifier names!
public class ABMICalculator
{
    public double calculateBMI(double weight, double height)
    {
        return weight/(height*height);
    }
}

public class ABMIC
{
    public double cBMI(double w, double h)
    {
        return w/(h*h);
    }
}
public class ABMICalculator
{
    public double calculateBMI(double weight, double height)
    {
        return weight/(height*height);
    }
}

// weight is in Kgs, height in metres
public class ABMICalculator
{
    public double calculateBMI(double weight, double height)
    {
        // Comment: Any code in the program removed by the compiler.
        return weight/(height*height);
    }
}

Single-line Comment: Begins with // and ends at line end.
class ABMICalculator
{
    double calculateBMI(double weight, double height)
    {
        return (height*height)/weight
    }
}

Errors

- Logic Error
- Syntax Error
- Access Error
- Semantics Error
Javac Syntax Error Reporting

ABMICalculator.java (3,3) : error J0232: Expected '{' or ';
ABMICalculator.java (3,3) : error J0021: Expected type specifier
ABMICalculator.java (3,3) : error J0019: Expected identifier
ABMICalculator.java (5,1) : error J0020: Expected 'class' or 'identifier'

public double calculateBMI(double height, double weight) {
    return weight/(height*height);
}
ECLIPSE: WE’VE COME A LONG WAY

Place (Hover) mouse on error

Click mouse on error
METHOD ACCESS ERROR

- You instantiate a ABMICalculator object

- but there is no ABMICalculator menu item

- Reason
  - You have not defined any public methods in ABMICalculator
USER ERROR

While instantiating a AXYZCalculator object

you get the following error

Reason
- AXYZCalculator class does not exist
**User Error**

- While instantiating a abmicalculator object

- you get the following error

- Reason
  - Class name is spelled with incorrect case
USER ERROR

- When invoking the Square method in ASquareCalculator

- you get the following error

- Reason
  - Parameter value is not in the function domain (set of allowed parameter values)
**SUMMARY: JAVA VS. REAL-WORLD**

<table>
<thead>
<tr>
<th>Java</th>
<th>Real-world</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Factory</td>
</tr>
<tr>
<td>Computer Object</td>
<td>Manufactured Physical Object</td>
</tr>
<tr>
<td>Method</td>
<td>Operation</td>
</tr>
<tr>
<td>Invoking/Executing a Method</td>
<td>Performing an Operation</td>
</tr>
<tr>
<td>Instance of a Class</td>
<td>Manufactured by a Factory</td>
</tr>
<tr>
<td>Defining/Declaring a Class</td>
<td>Constructing a Factory</td>
</tr>
<tr>
<td>Instantiating a Class</td>
<td>Manufacturing an Object</td>
</tr>
<tr>
<td>Class (Static) Method</td>
<td>Operation on a Factory</td>
</tr>
<tr>
<td>Main Class Method</td>
<td>Initiates computation</td>
</tr>
<tr>
<td>Instance (Non static) method</td>
<td>Operation on an instance of a class</td>
</tr>
<tr>
<td>Instance (Non static) method</td>
<td>Grouping of factories by states, country</td>
</tr>
</tbody>
</table>
EXTRA SLIDES
public class ASquareCalculator
{
    public int square(int x) 
    {
        return x*x;
    }
}

public class SquareCalculatorDriver
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new ASquareCalculator();
        System.out.println (squareCalculator.square(5));
    }
}
Rerun Program

How to not re-run program without writing tedious UI code?

```java
public class ASquareCalculator {
    public int square(int x) {
        return x*x;
    }
}
```

```java
public class SquareCalculatorDriver {
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new ASquareCalculator();
        System.out.println (squareCalculator.square(Integer.parseInt(args[0])));
    }
}
```
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}

class SquareCalculatorDriver
{
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new ASquareCalculator();
        System.out.println (squareCalculator.square(341));
    }
}
PROGRAMMED VS. INTERACTIVE CALL

**Programmed Call**
bmiCalculator.calculateBMI(75, 1.77);
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}

public class SquareCalculatorDriver
{
    public static void main (String[] args)
    {
        System.out.println ( (new ASquareCalculator()).square(5) );
    }
}
CHANGING PARAMETER

```java
public class ASquareCalculator {
    public int square(int x) {
        return x * x;
    }
}
```

```java
public class SquareCalculatorDriver {
    public static void main (String[] args) {
        System.out.println ((new ASquareCalculator()).square(341));
    }
}
```

Must change code and re-run program
Invoking a Method and Viewing the Result
**FORMAL VS. ACTUAL PARAMETERS**

```java
public class ABMICalculator {
    public double calculateBMI(double weight, double height) {
        return weight/(height*height);
    }
}
```

Invoke `calculateBMI`

Assigned variables

Actual Parameters

Formal Parameters

<table>
<thead>
<tr>
<th>variables</th>
<th>memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>weight</td>
<td>74.98</td>
</tr>
<tr>
<td>height</td>
<td>1.94</td>
</tr>
</tbody>
</table>
public class ABMICalculator
{
    public double calculateBMI(double weight, double height)
    {
        return weight/(height*height);
    }
}

public class BMICalculatorDriver
{
    public static void main (String[] args) {
        ABMICalculator bmiCalculator = new ABMICalculator();
        System.out.println (bmiCalculator.calculateBMI(75, 1.77));
    }
}
**Instance vs. Class (Static) Method**

```
public class ABMICalculator {
    public double calculateBMI(double weight, double height) {
        return weight / (height * height);
    }
}
```

```
public class BMICalculatorDriver {
    public static void main(String[] args) {
        ABMICalculator bmiCalculator = new ABMICalculator();
        System.out.println(bmiCalculator.calculateBMI(75, 1.77));
    }
}
```
Program Objects ~ Manufactured Objects

Blue print → created from → Object

Operations
- accelerate
- brake

Class
- Program Object
  - instance of

Methods
- add
- subtract

Program Object
- execute
- invoke
- call
**PROGRAM OBJECTS ~ MANUFACTURED OBJECTS**

Class is an object with dynamic state on which you can invoke static methods and can have multiple classes producing same kind of object ~ factory instead of blueprint.
Parameter and return types are integers

But height (m) and weight (kg) are expressed as decimals

How do we solve the discrepancy?
package bmi;
public class ABMICalculator {
    public int calculateBMI(int weight, int height) {
        return weight/(height*height);
    }
}
PROGRAMMATIC VS. INTERACTIVE METHOD INVOCATION

(new ASquareCalculator()).square(5);

Target Object  Method Name  Parameter

Invoking operations is essentially form filling
**Object Variables**

```java
public class ASquareCalculator {
    public int square(int x) {
        return x*x;
    }
}
```

```java
public class SquareCalculatorUser {
    public static void main (String[] args) {
        ASquareCalculator squareCalculator = new ASquareCalculator();
        System.out.println(squareCalculator.square(5));
    }
}
```