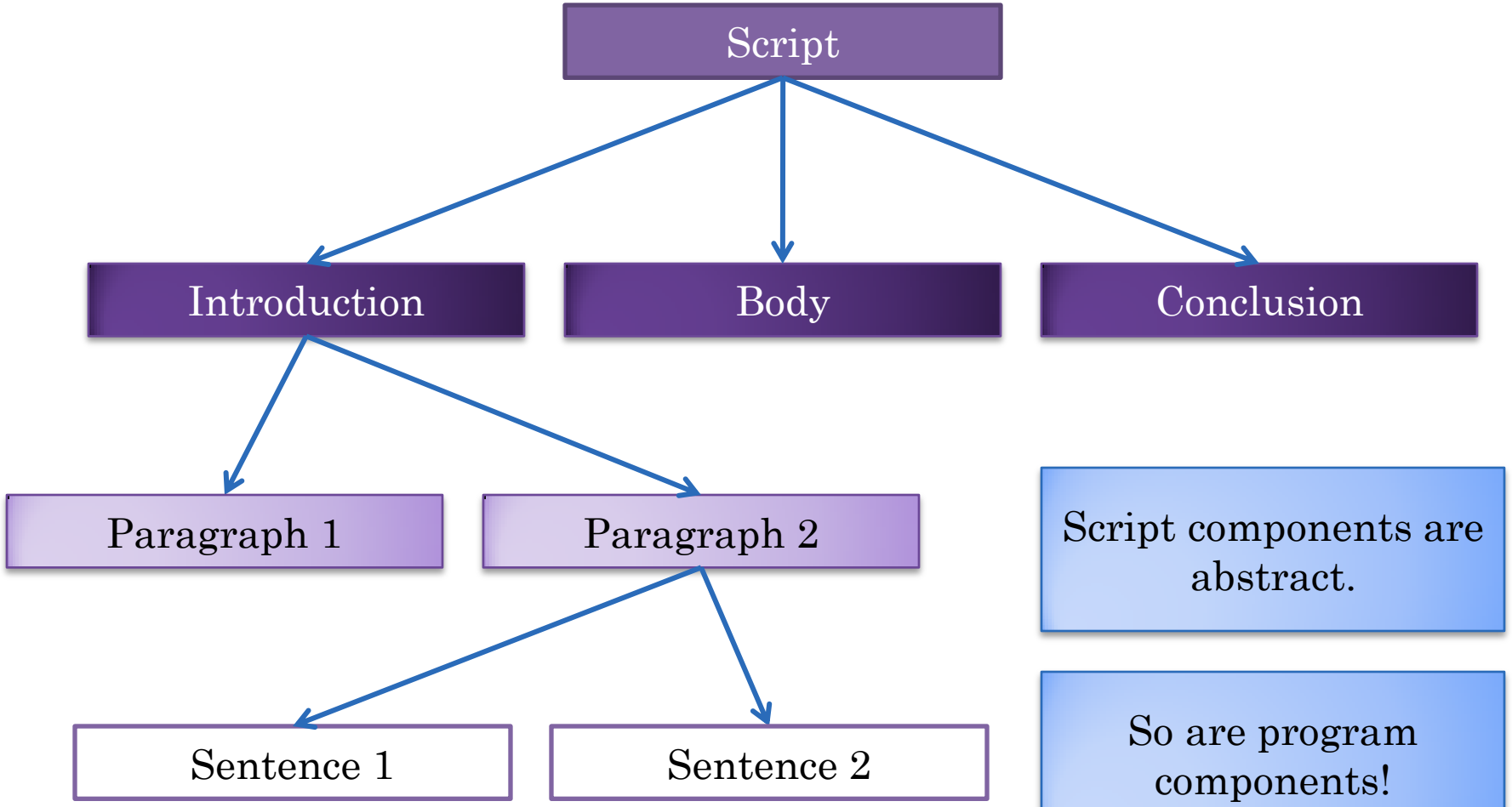
COMP 110/401
OBJECTS

Instructor: Prasun Dewan

Prerequisites

- Theater (optional)

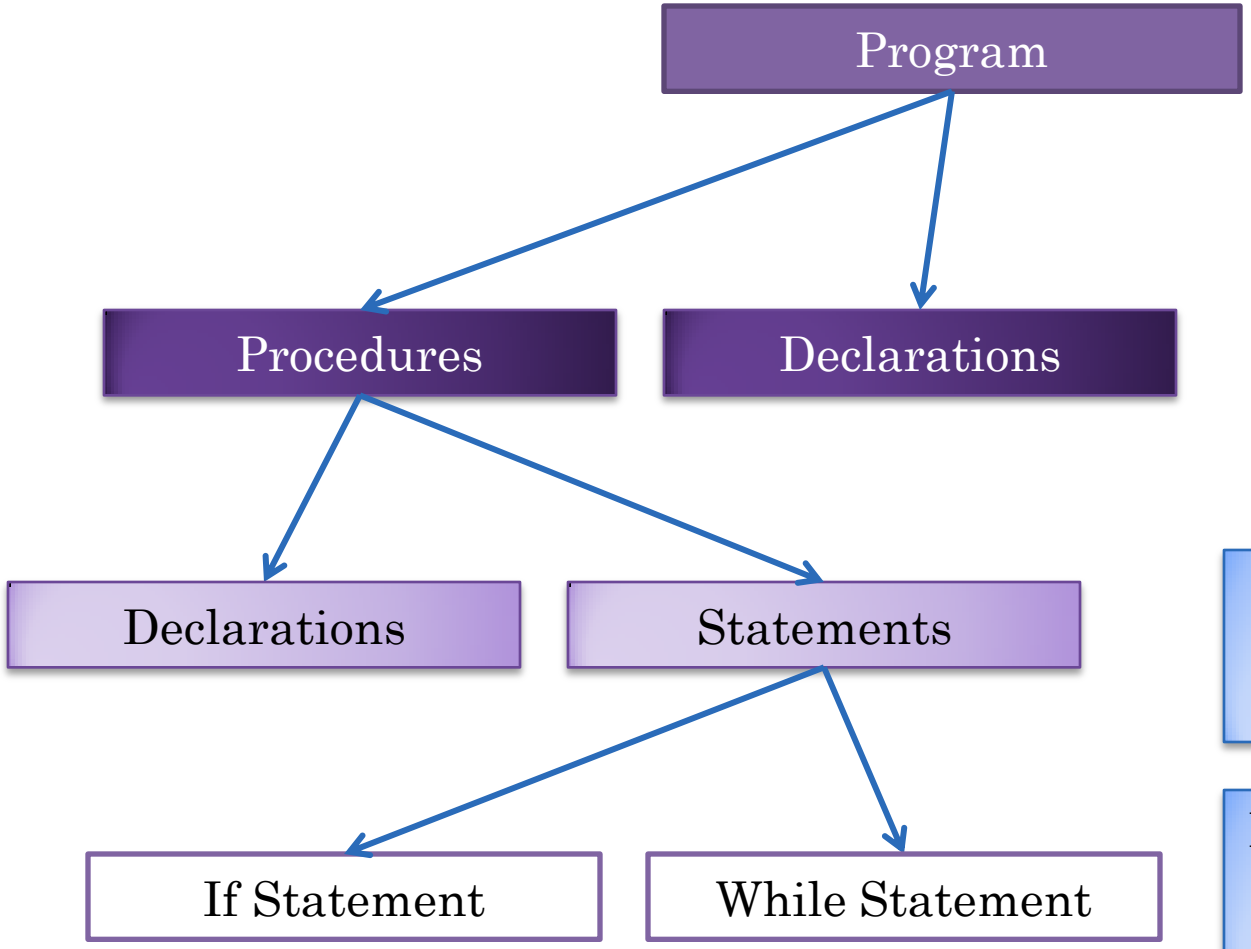
Structuring in Scripts



Script components are abstract.

So are program components!

Conventional Languages



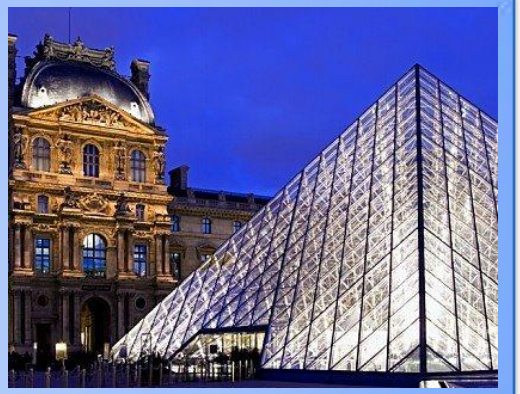
Script analogy works for conventional programming

For O-O programming analogy is physical objects

Program Components ~ Physical Objects

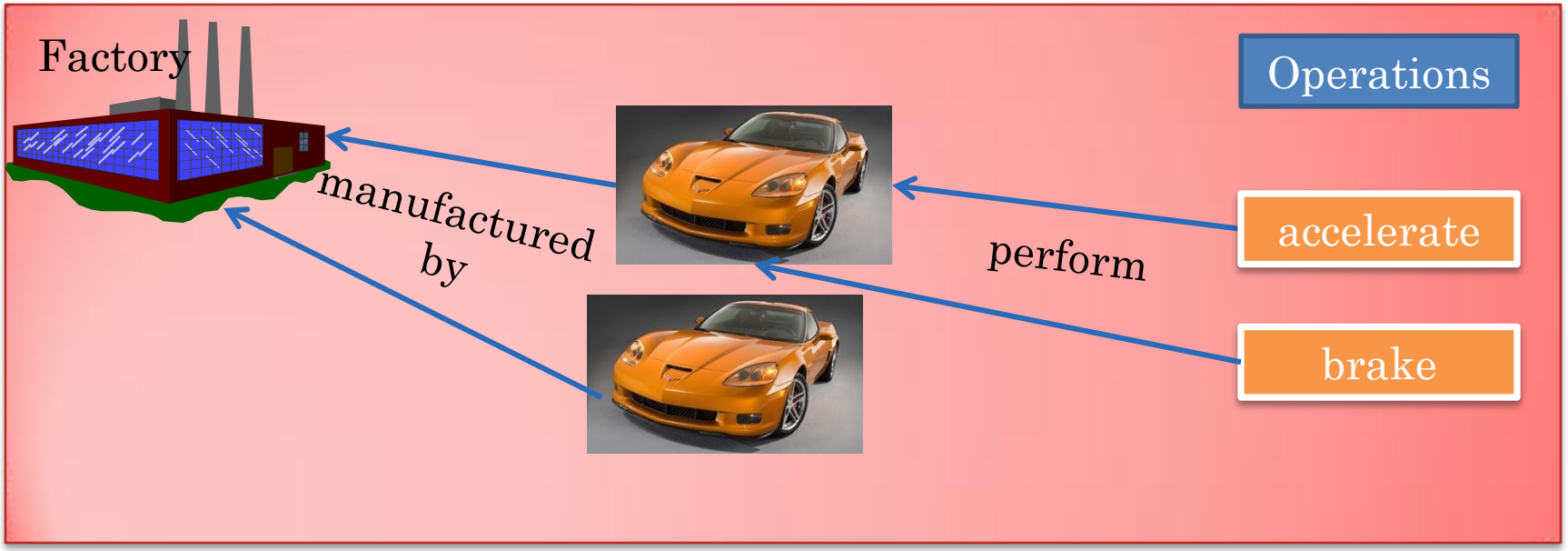
Natural Objects

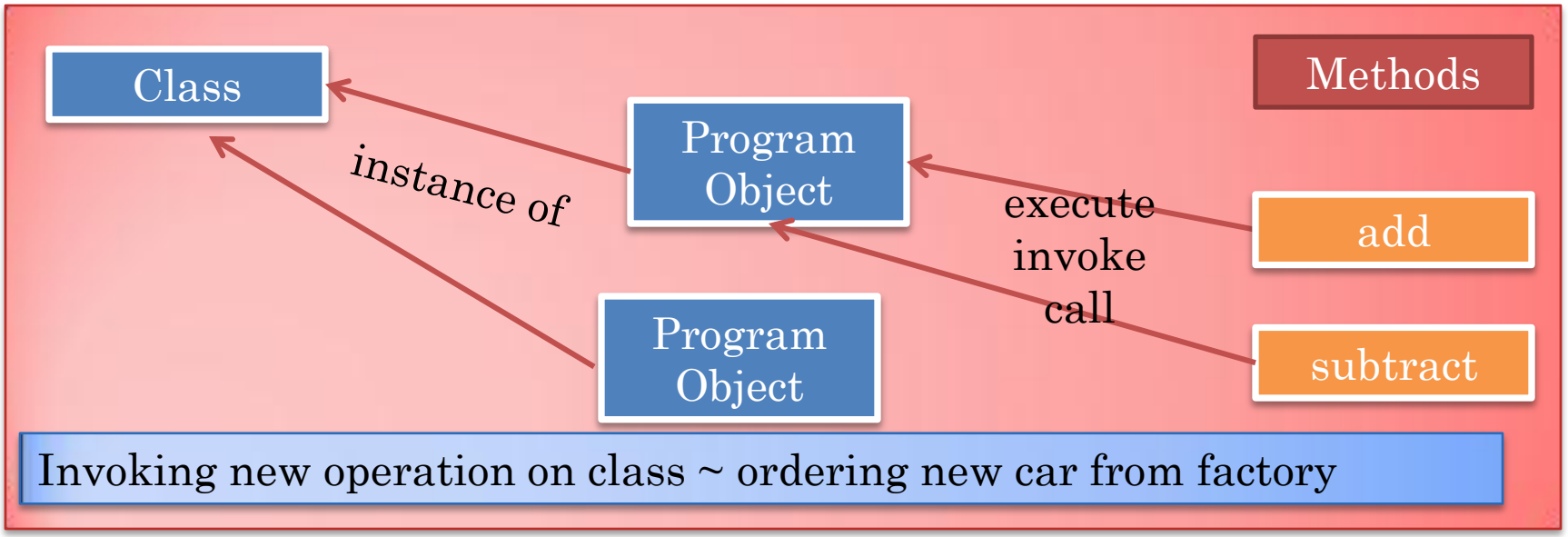
Manufactured Objects



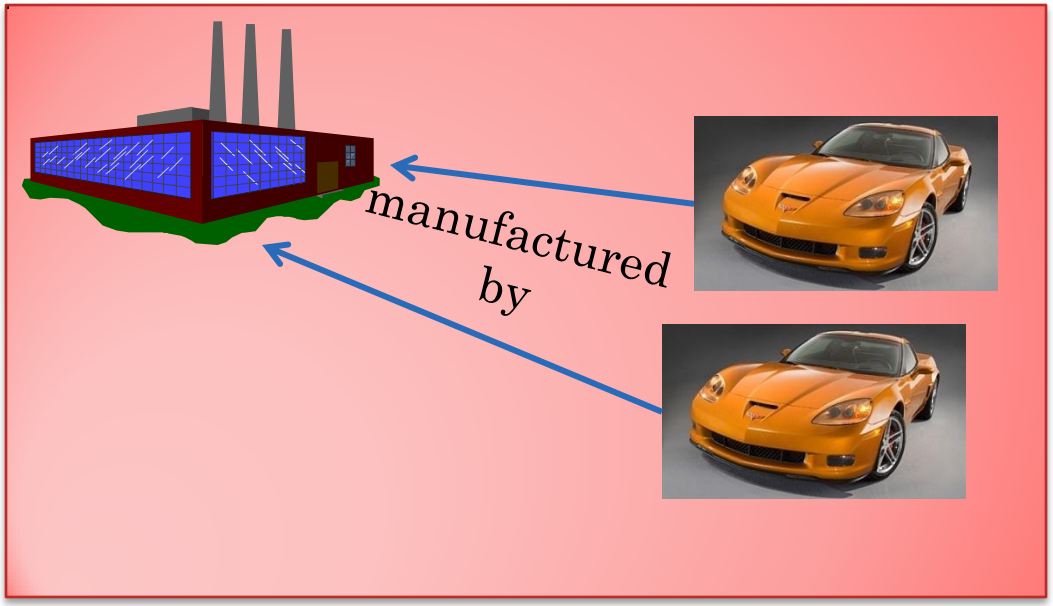
~ Program Components

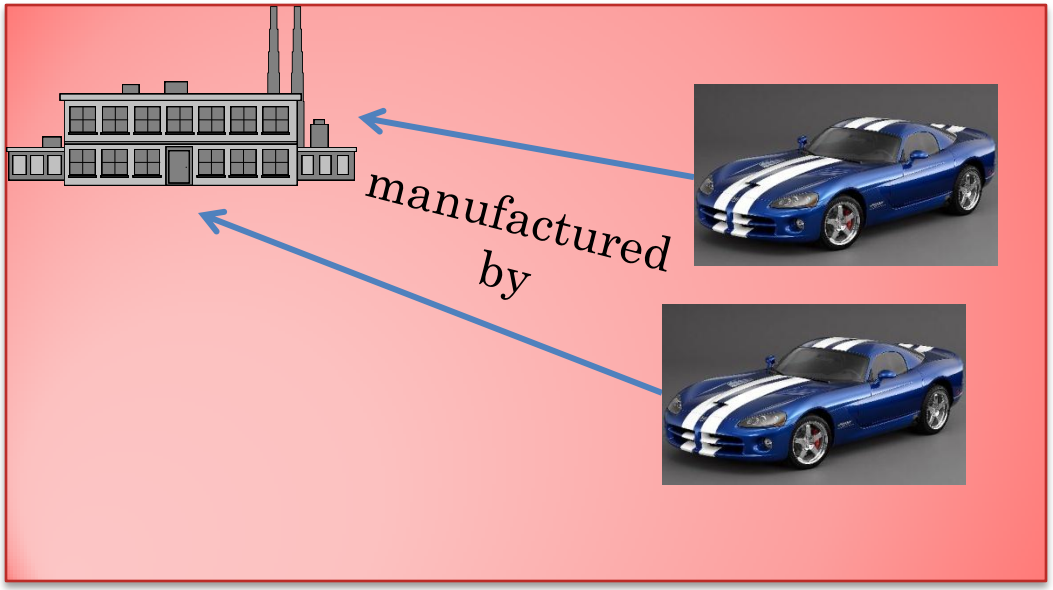
Program Objects ~ Manufactured Objects



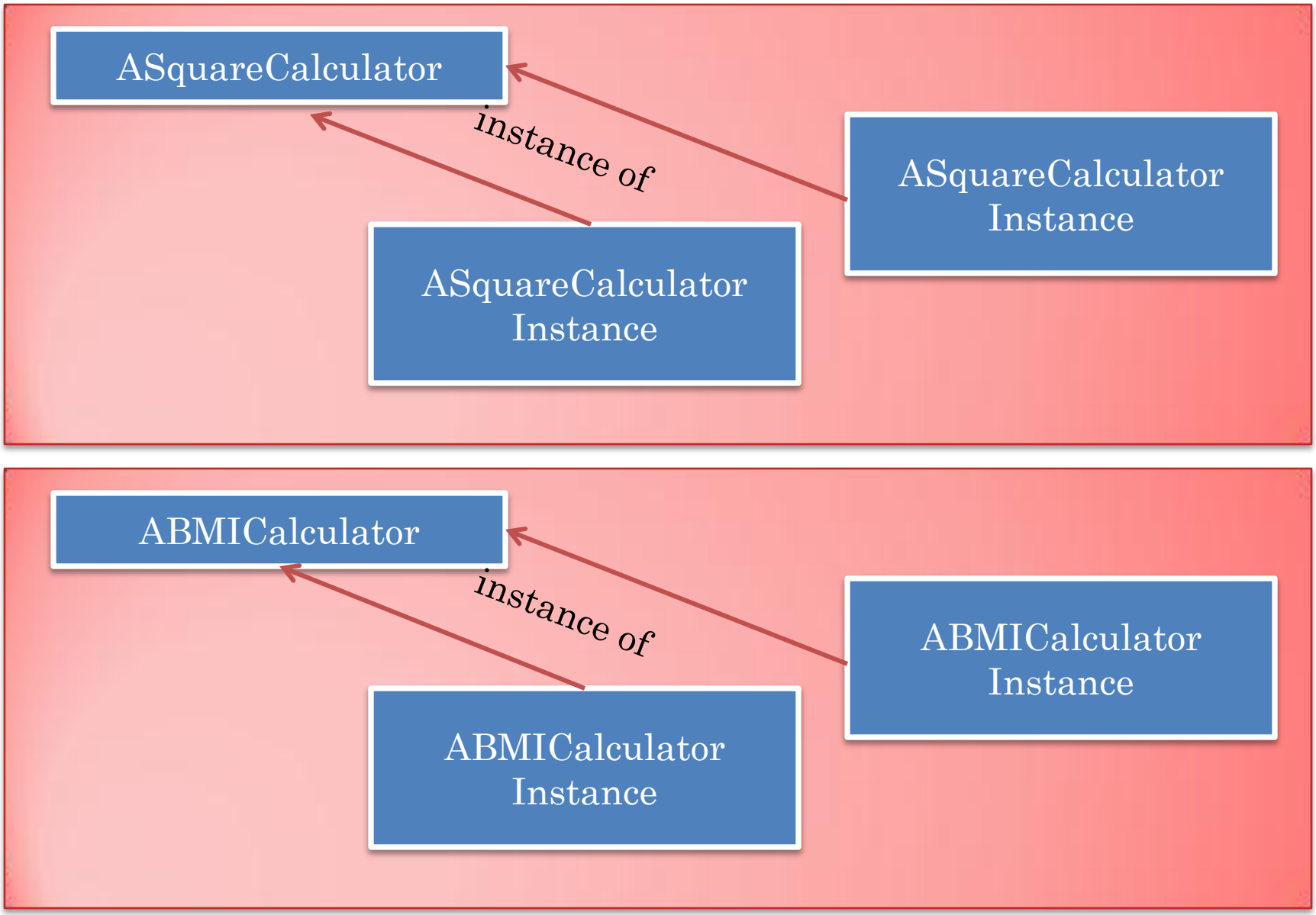


Classification Through Factories





Classification Through Classes



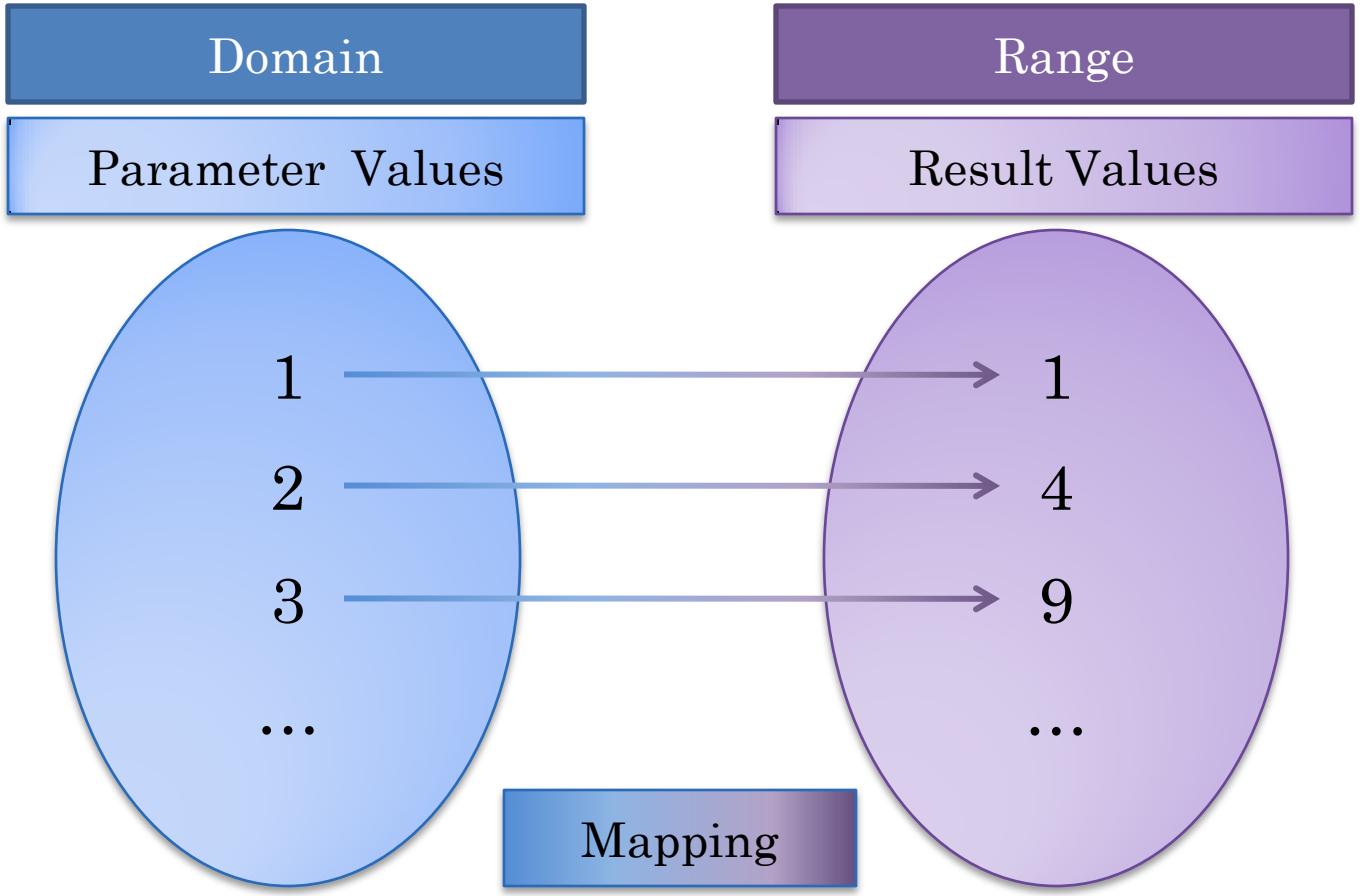
A Simple Class: ASquareCalculator

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

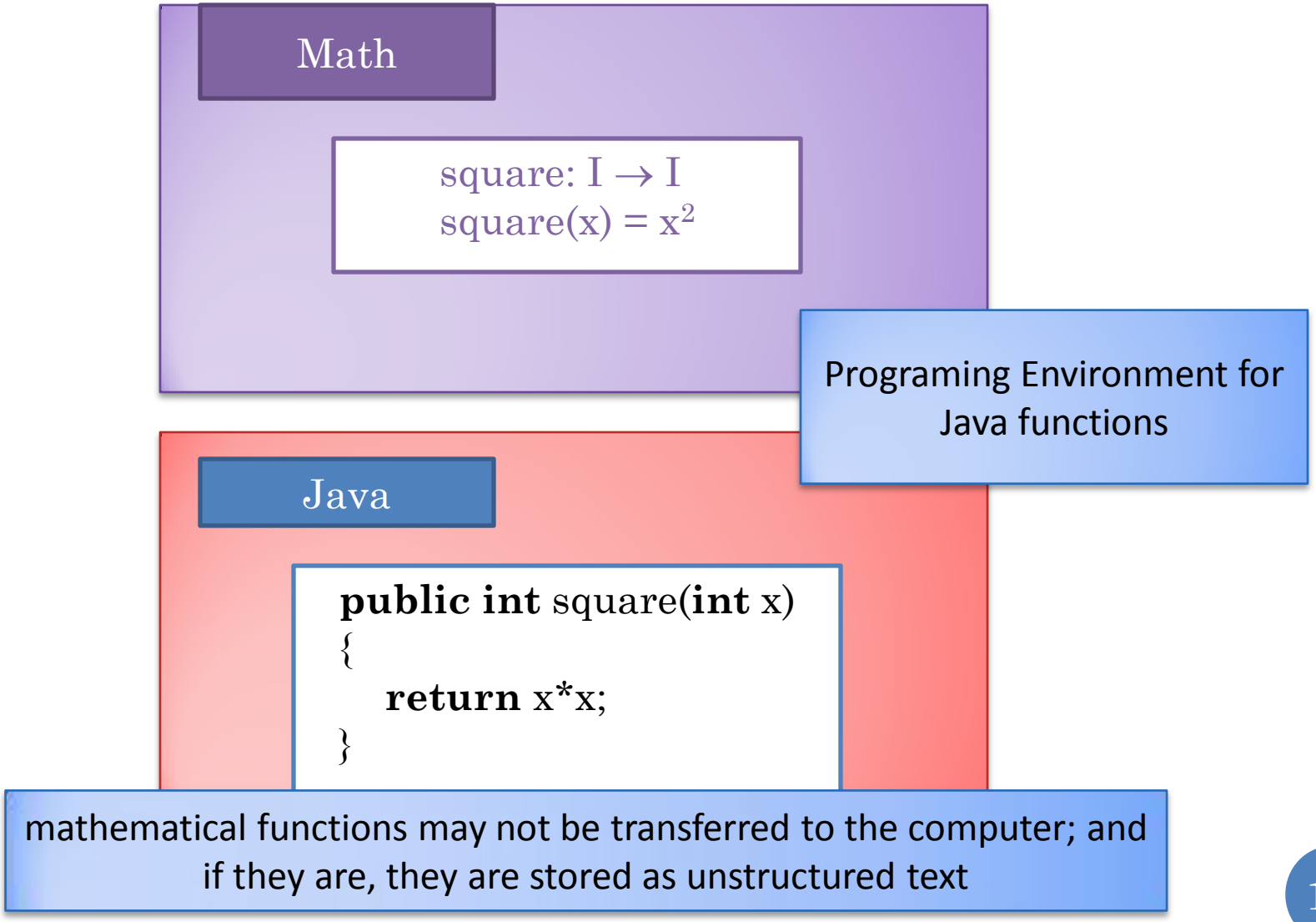
No static because operation on instance of ASquareCalculator rather than on the class ASquareCalculator

Methods, parameters and return value?

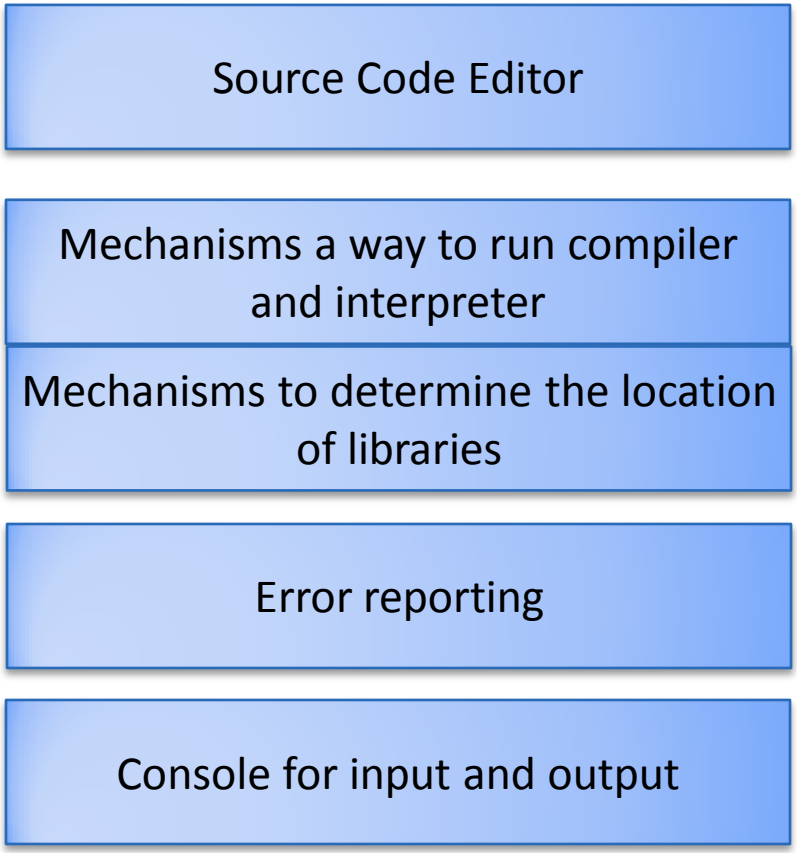
Nature of a Function



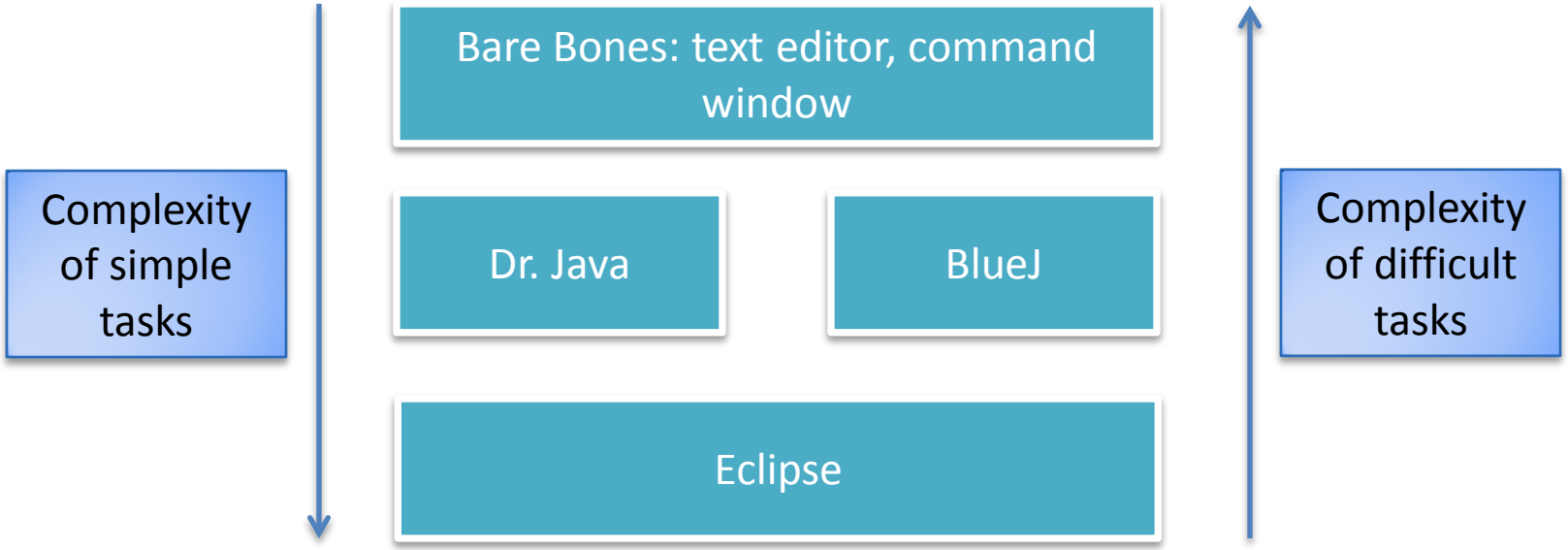
Mathematics vs. Java Function Syntax



Programming Environment



Example Programming Environments



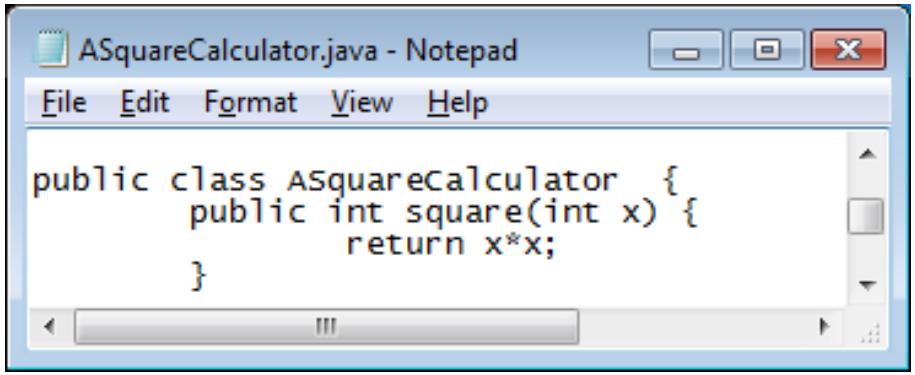
Free to use any programming environment

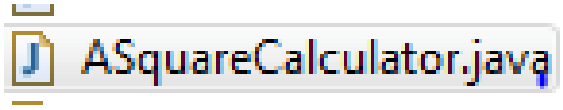
We will provide help with Eclipse

Stored Where?

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

Saving Class in a File



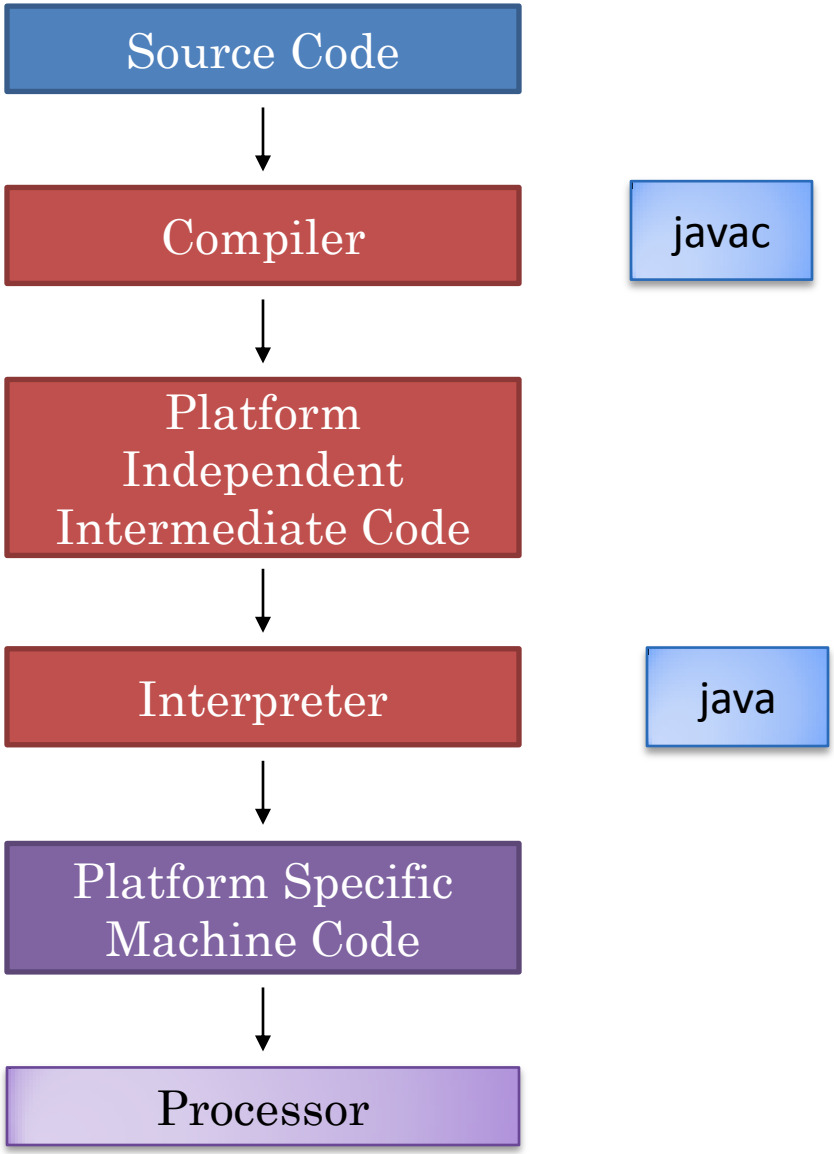



```
public class ASquareCalculator {
    public int square(int x) {
        return x*x;
    }
```

Class C is saved in a file called C.Java

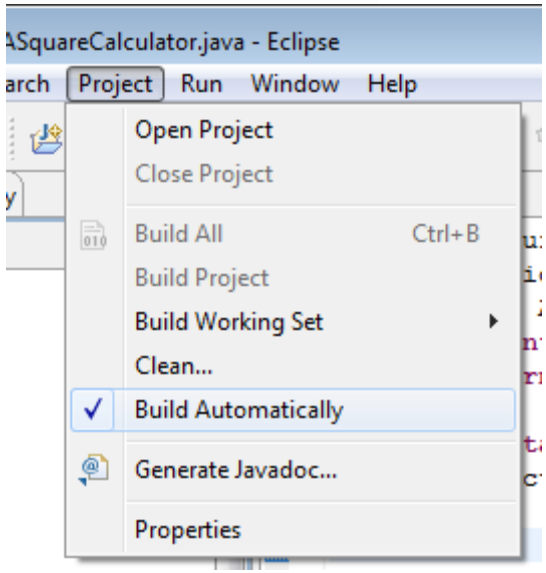
Programming environment may automatically create file name from class name

Java Translation/Execution Process



Compiling a Class

ASquareCalculator.java 3/21/2012 11:04 AM JAVA File

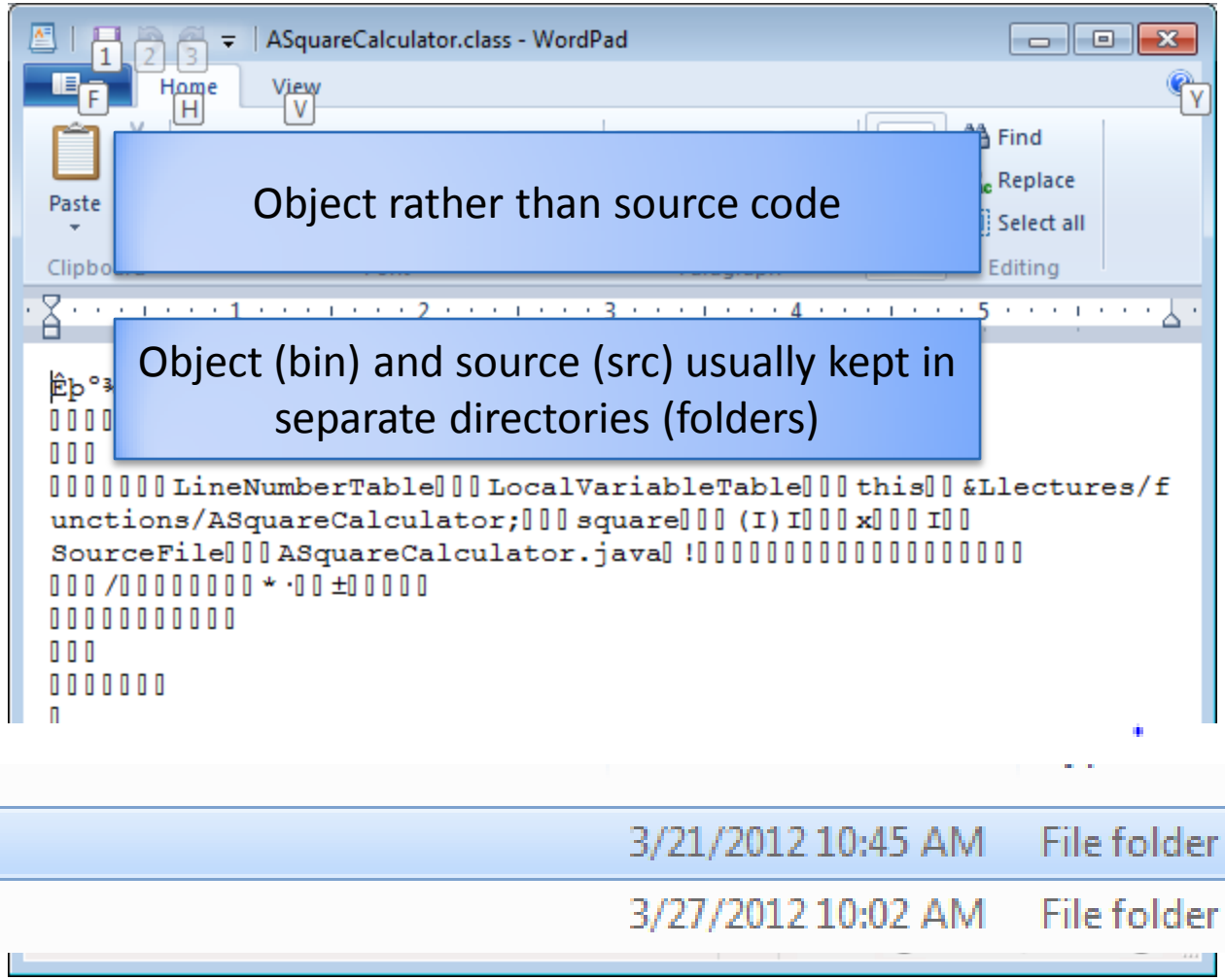


Command window or programming environment

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

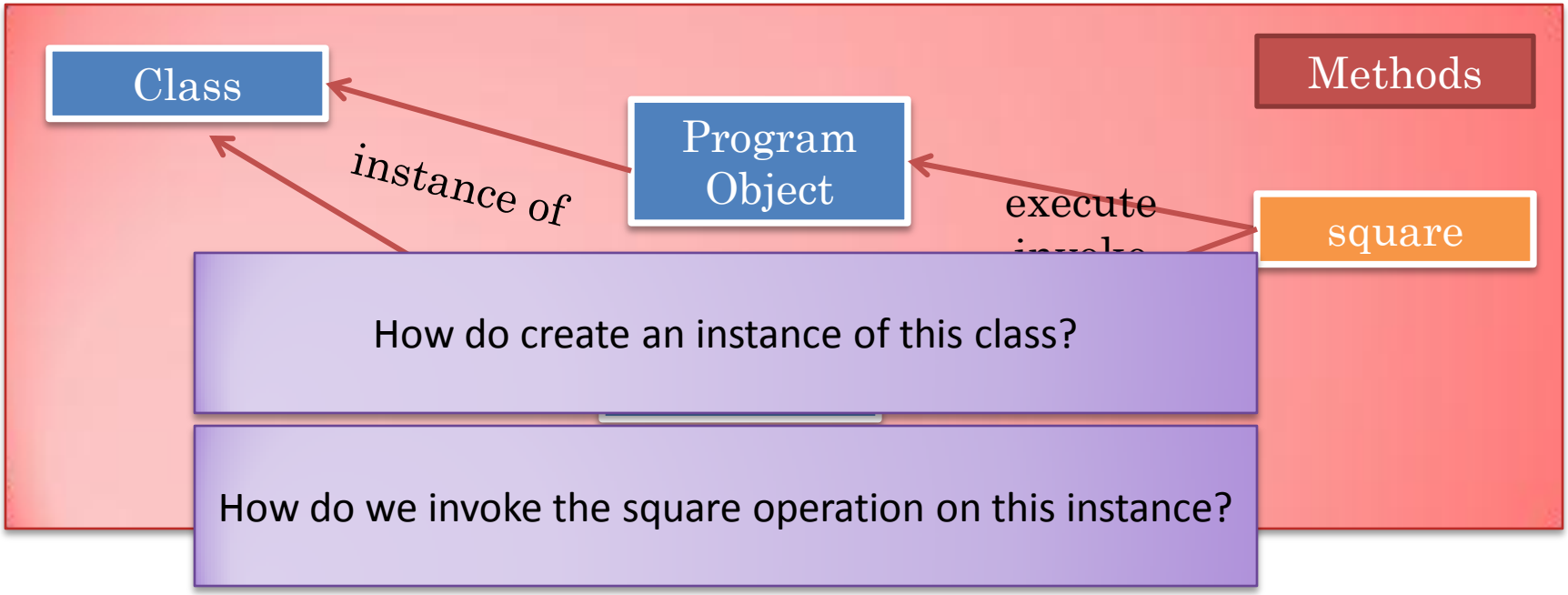
ASquareCalculator.class 3/21/2012 11:04 AM CLASS File

Class File

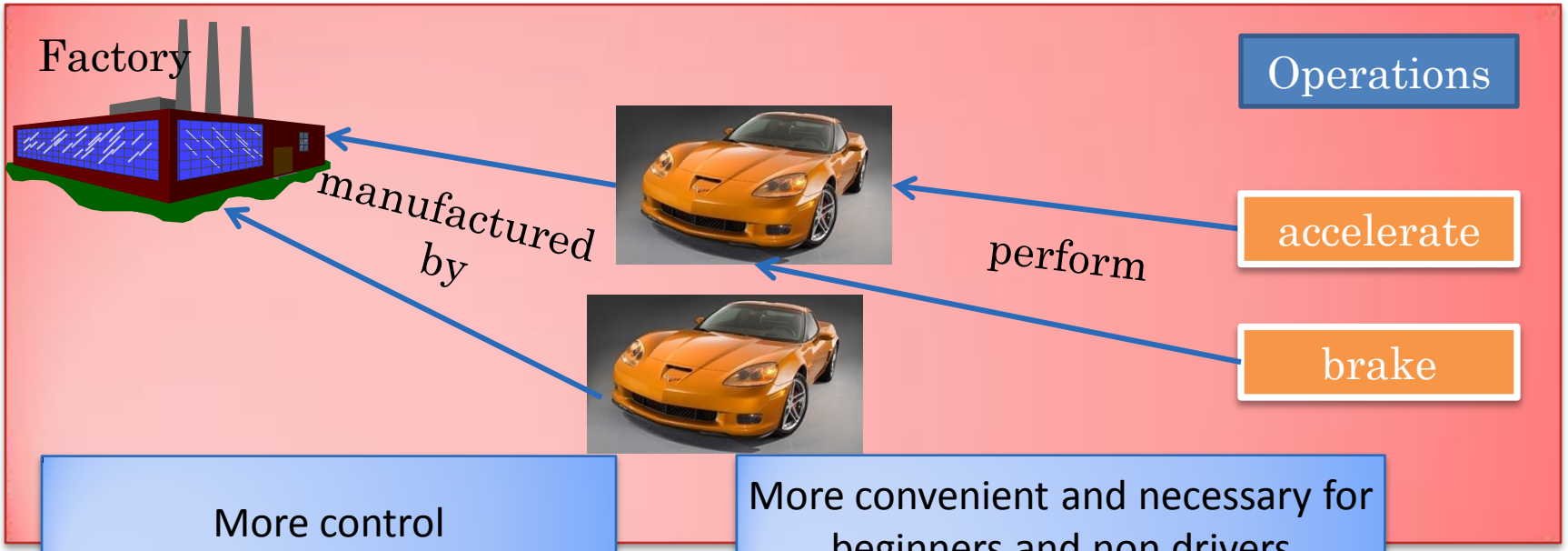

ASquareCalculator.class - WordPad
Home
View
Paste
Find
Replace
Select all
Editing
Object rather than source code
Object (bin) and source (src) usually kept in separate directories (folders)
LineNumberTable
LocalVariableTable
this
&Llectures/functions/ASquareCalculator;
square
(I)I
x
I
SourceFile
ASquareCalculator.java
bin
3/21/2012 10:45 AM
File folder
src
3/27/2012 10:02 AM
File folder

Using ASquareCalculator (Object Code)

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



Ordering and Using a Manufactured Object



More control

You can do it yourself

More convenient and necessary for beginners and non drivers

You can give orders to your chauffeur

Manual Driving: Writing our own Main

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

Object Creation

Invoked on class defining main

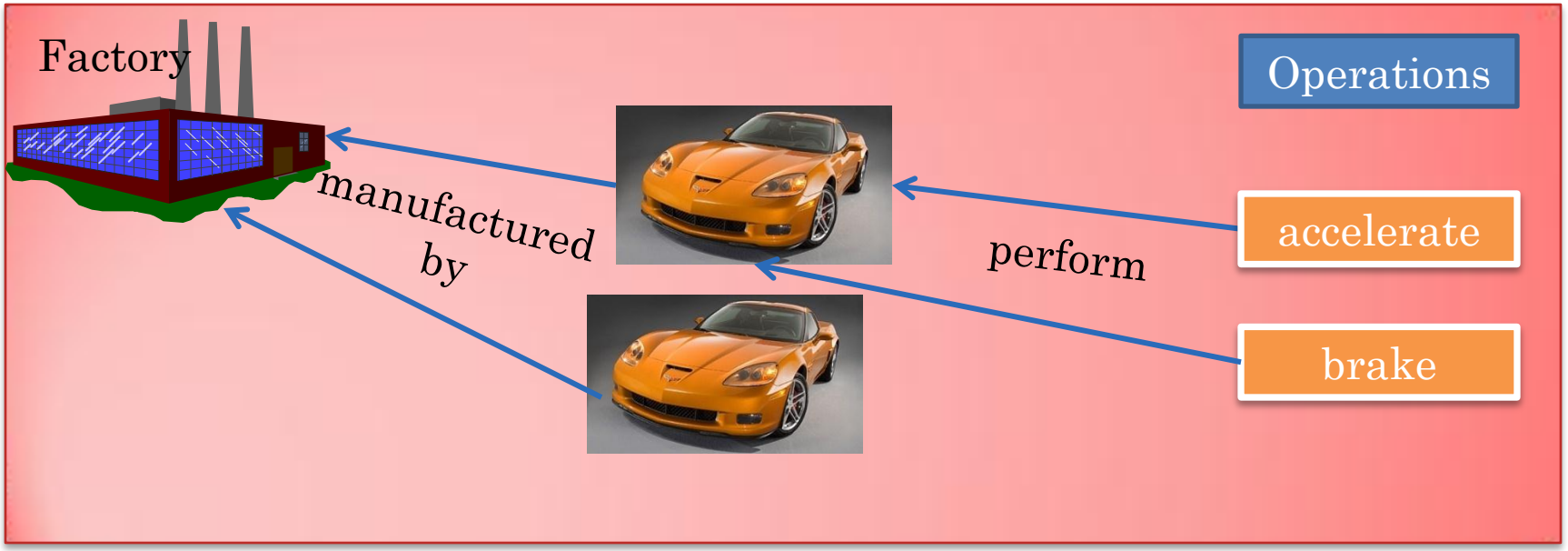
Magic for beginners

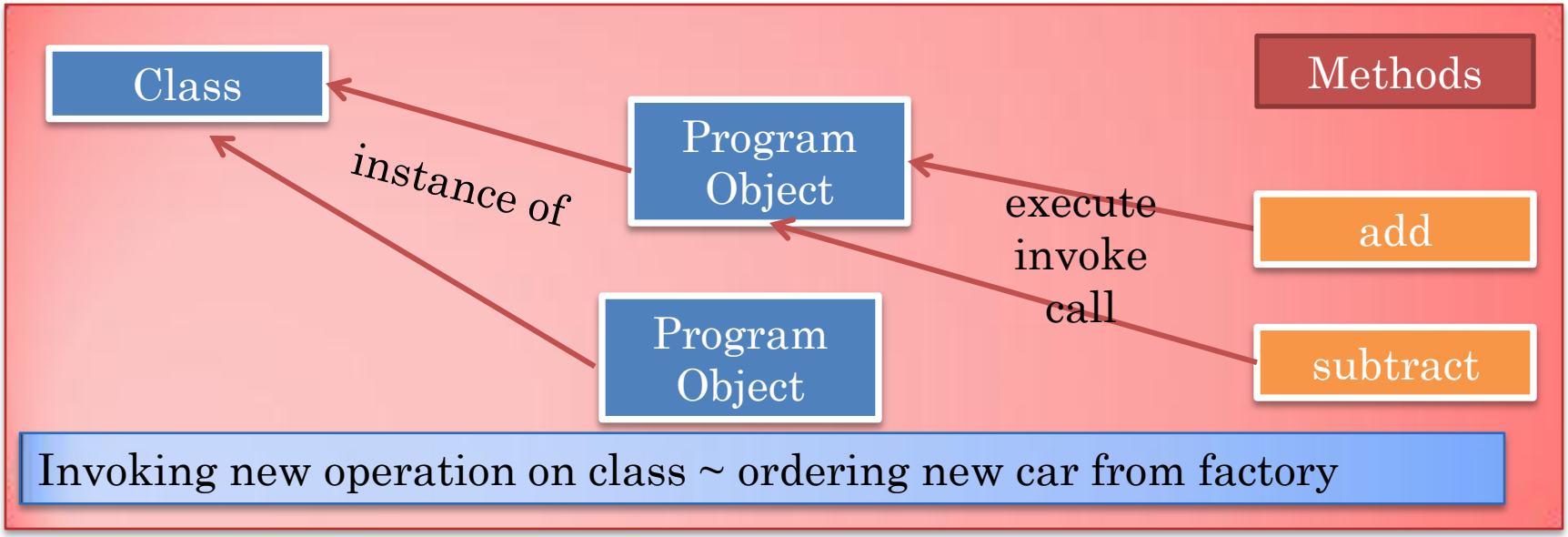
```
public class SquareCalculatorDriver
{
  public static void main (String[] args)
  {
     System.out.println (
        (new ASquareCalculator()).square(5)
     );
}
```

Operation invocation

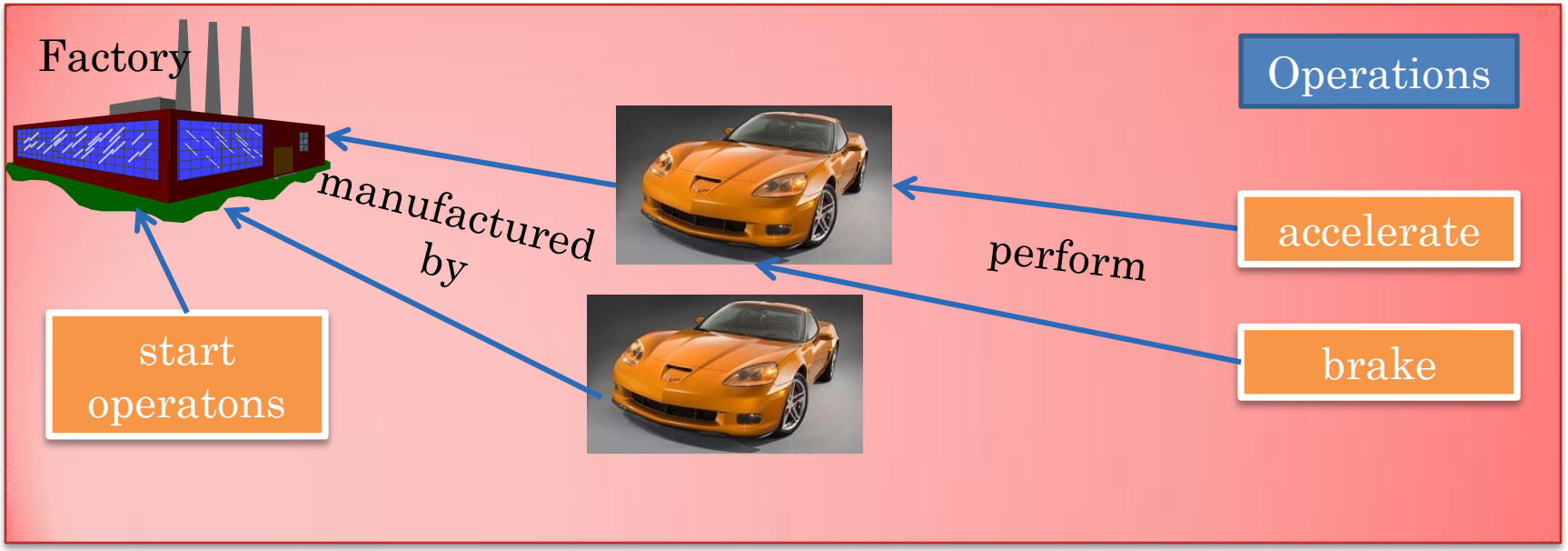
We can now "run" main to" start operations on the factory"

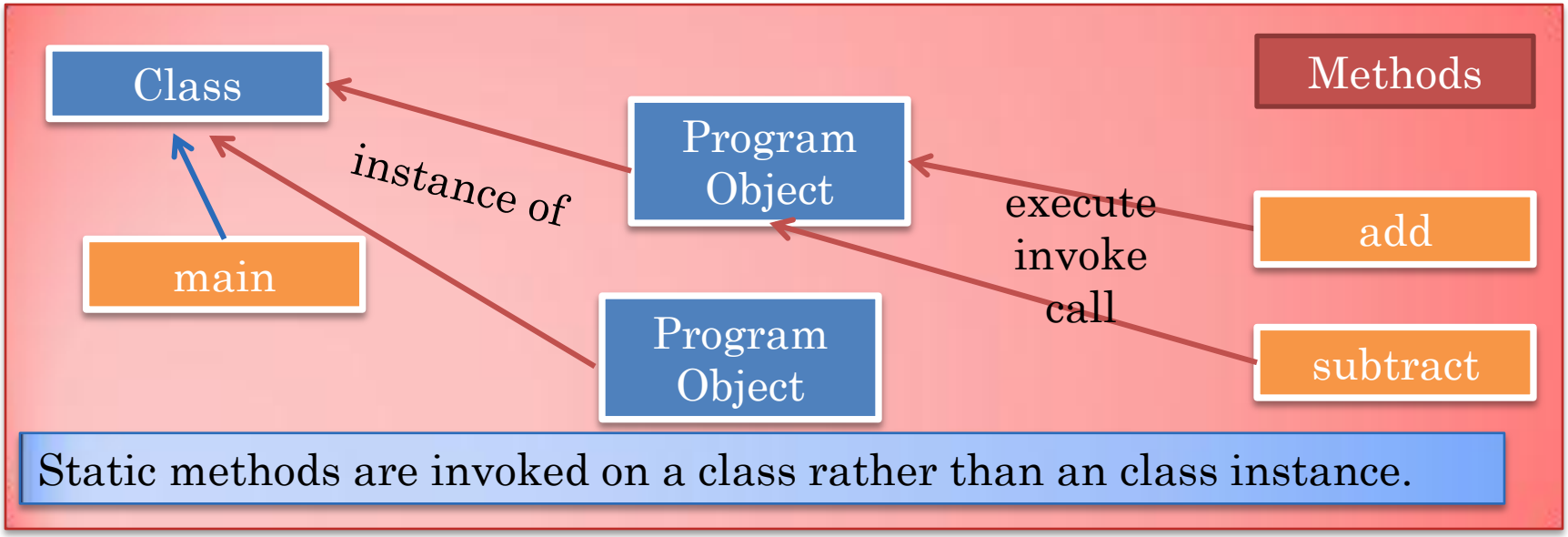
Analogy for instance Methods





Analogy for Static (Class) Methods





Calling Main from Our Program

```
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);
}
```

Main Method Processing Arg

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

Method invoked on Class Integer

```
public class SquareCalculatorDriver Driver{
  public static void main (String[] args)  {
            String myArgs[] = {"5"};
            SquareCalculatorDriver .main(myArgs);
}
```

Reading Int

```
int product = 1;
int nextNum = Console.readInt();
while (nextNum >= 0) {
        product = product* nextNum;
        nextNum = Console.readInt();
}
System.out.print (product);
```

Console Static Methods

```
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
}
```

Class with no instance methods

Real Life Analogy

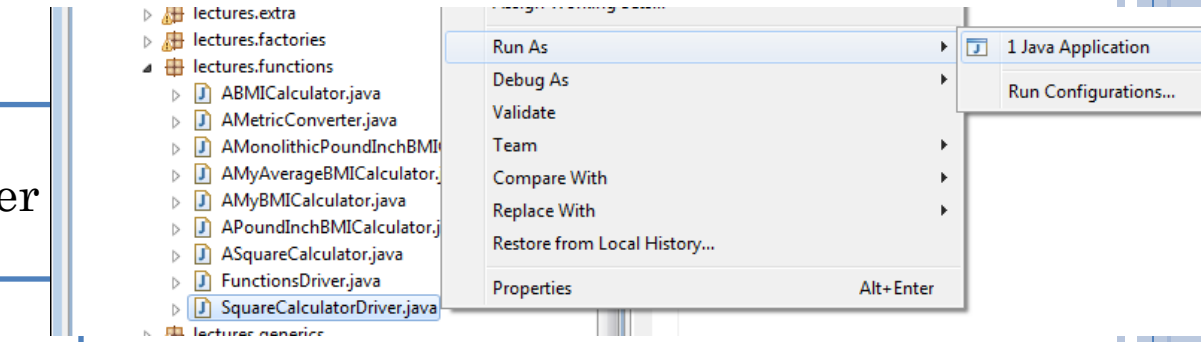
Warehouse, dealership

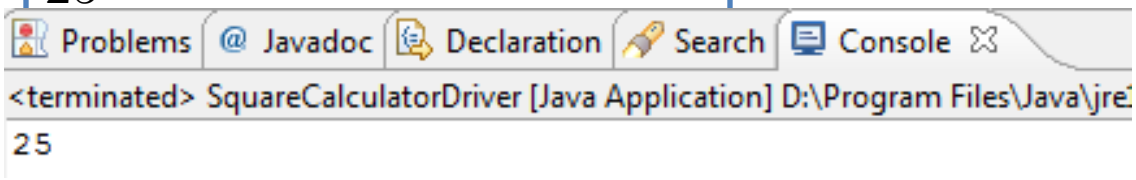
Running Your Own Main

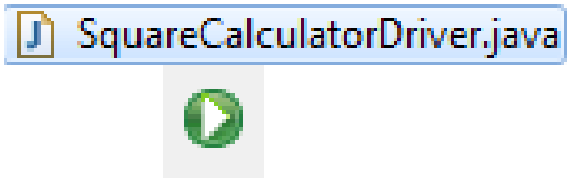
Command window or programming environment

```
java SquareCalculatorDriver
```

```
25
```

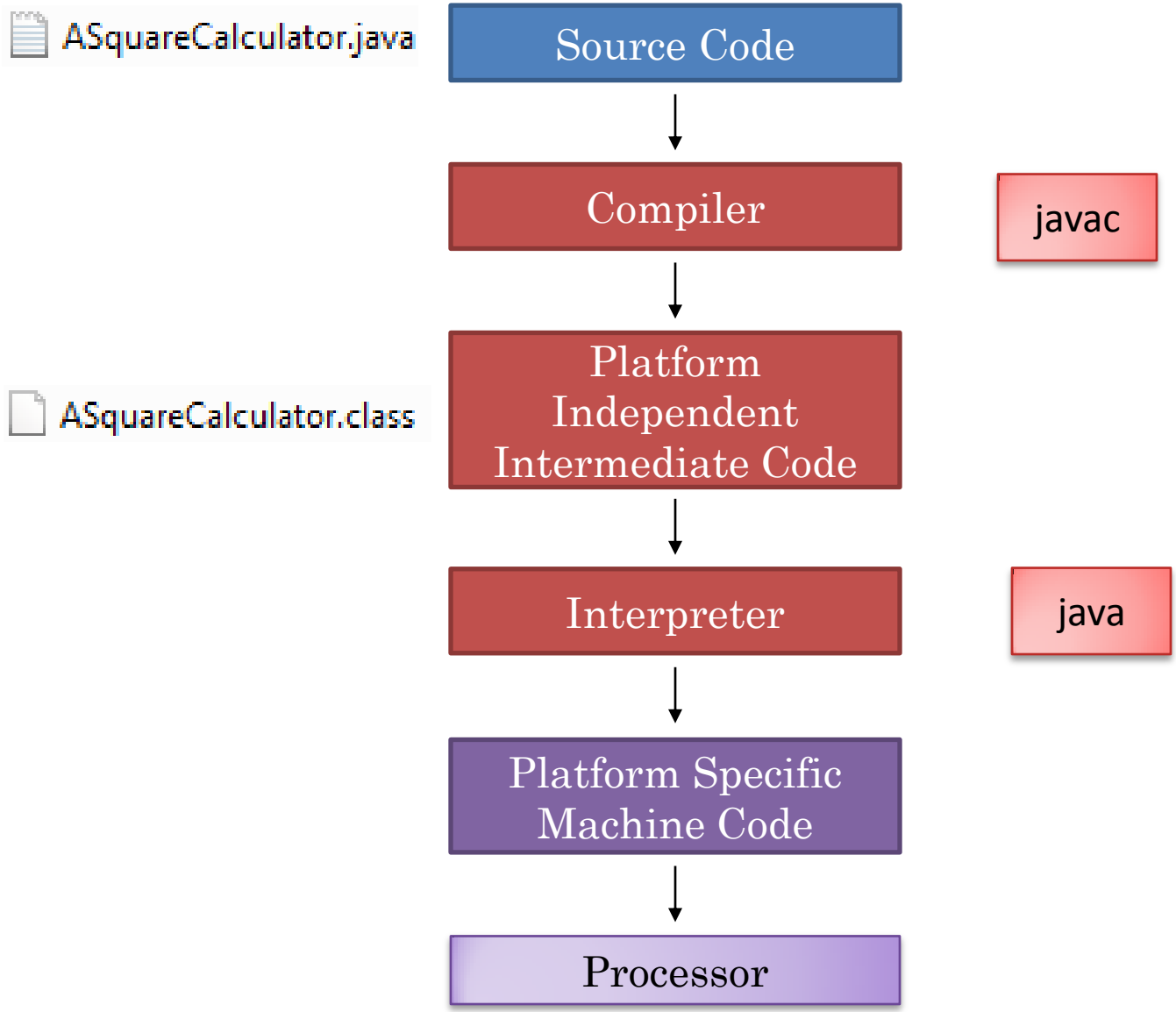




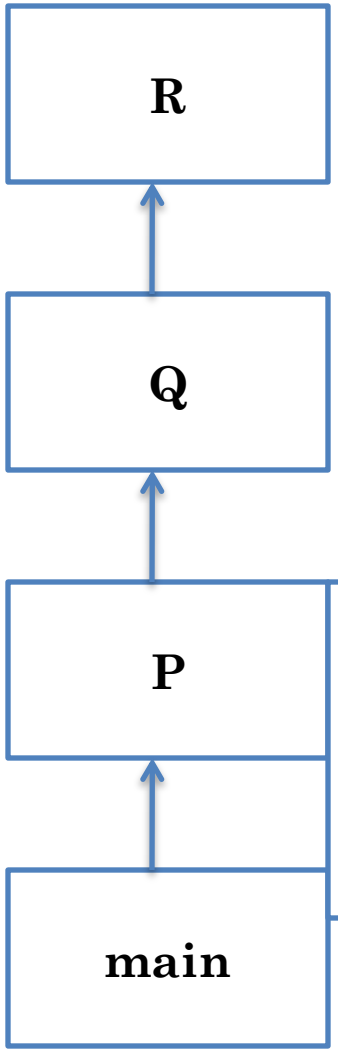



```
public class SquareCalculatorDriver
{
  public static void main (String[] args)
  {
     System.out.println (
        (new ASquareCalculator()).square(5)
     );
}
```

Java Translation Process



Main Method Details



Main method has predefined header.

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

All methods must be in some "class"

```
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.

Alternative Mains

```
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])));
}
```

Changing Parameter

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

Calculates 5*5

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

Changing Parameter

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

Calculates 341*341

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

For different parameter values must modify, compile and run program

Main Arg

101 students must understand arrays

Must run program multiple times

```
public class ASquareCalculator

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

Can get value from user as a main arg

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

User Input

Must run program multiple times

Unless we write tedious loops with prompting

110 students do not know loops

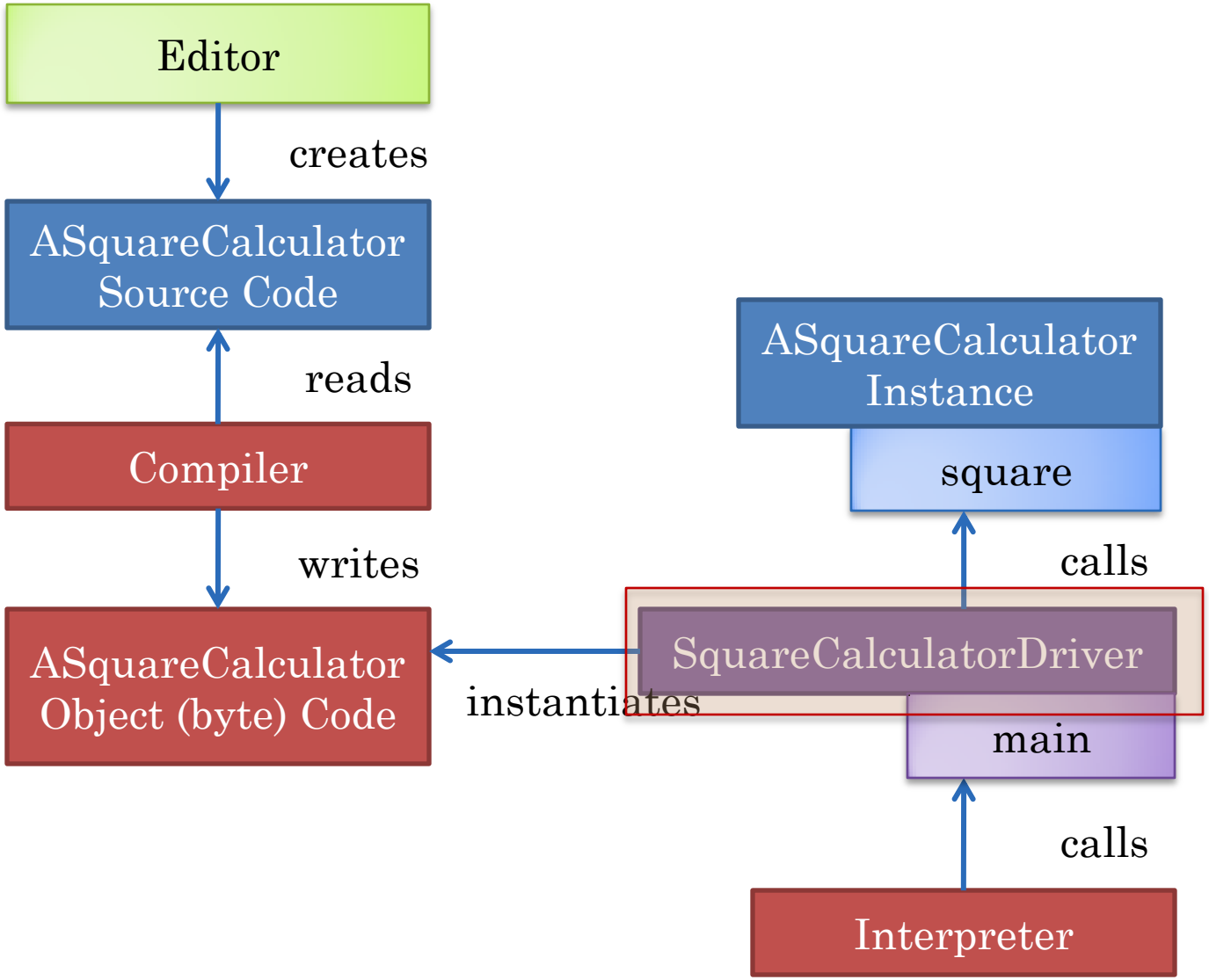
```
class ASquareCalculator

public int square(int x)

return x*x;
```

Can get value from user

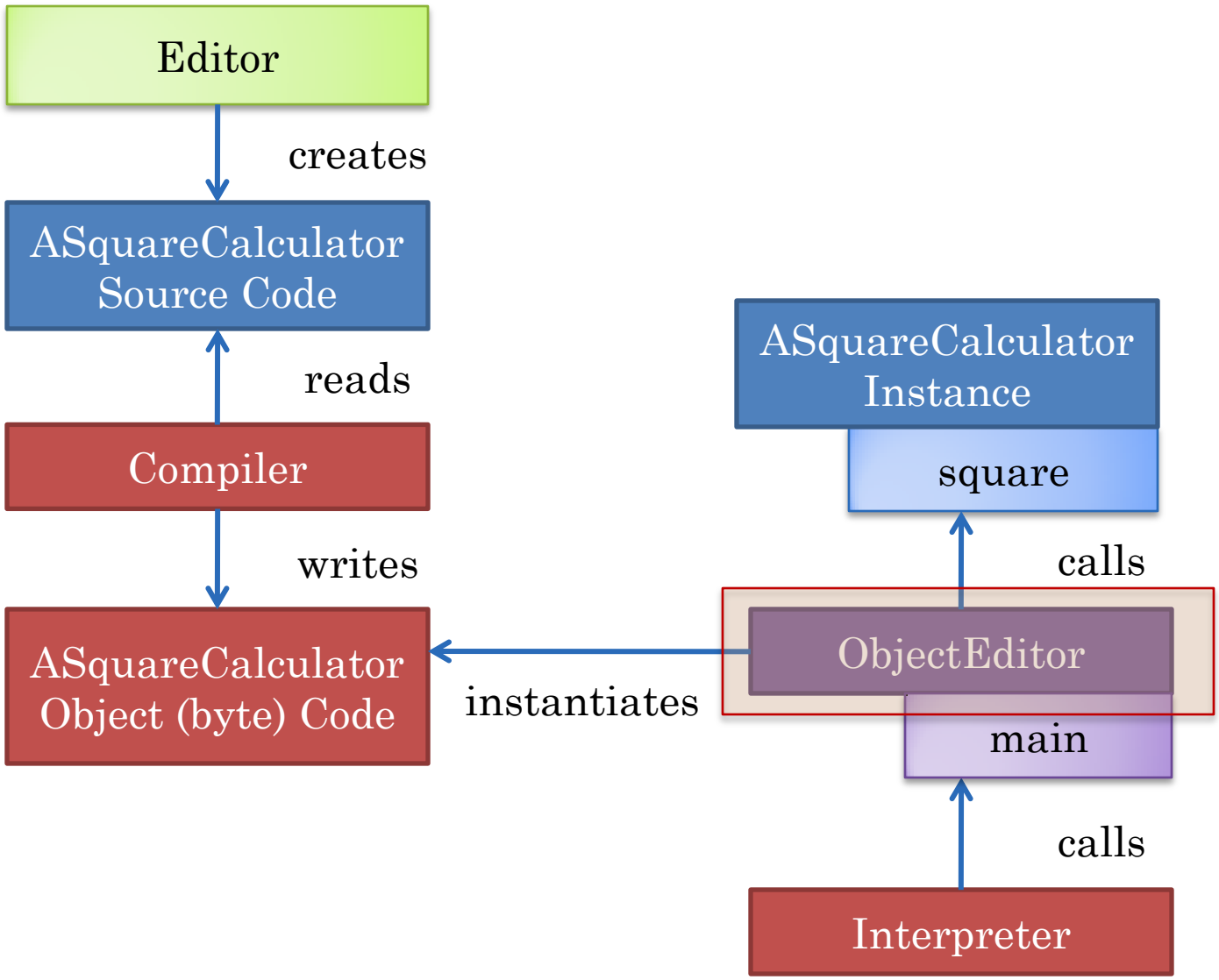
```
public class SquareCalculatorDriver
{
  public static void main (String[] args)
  {
     System.out.println (
        (new ASquareCalculator(Console.readInt())));
}
```

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

Self Driven Program Development



Chauffeur-Driven Program Development



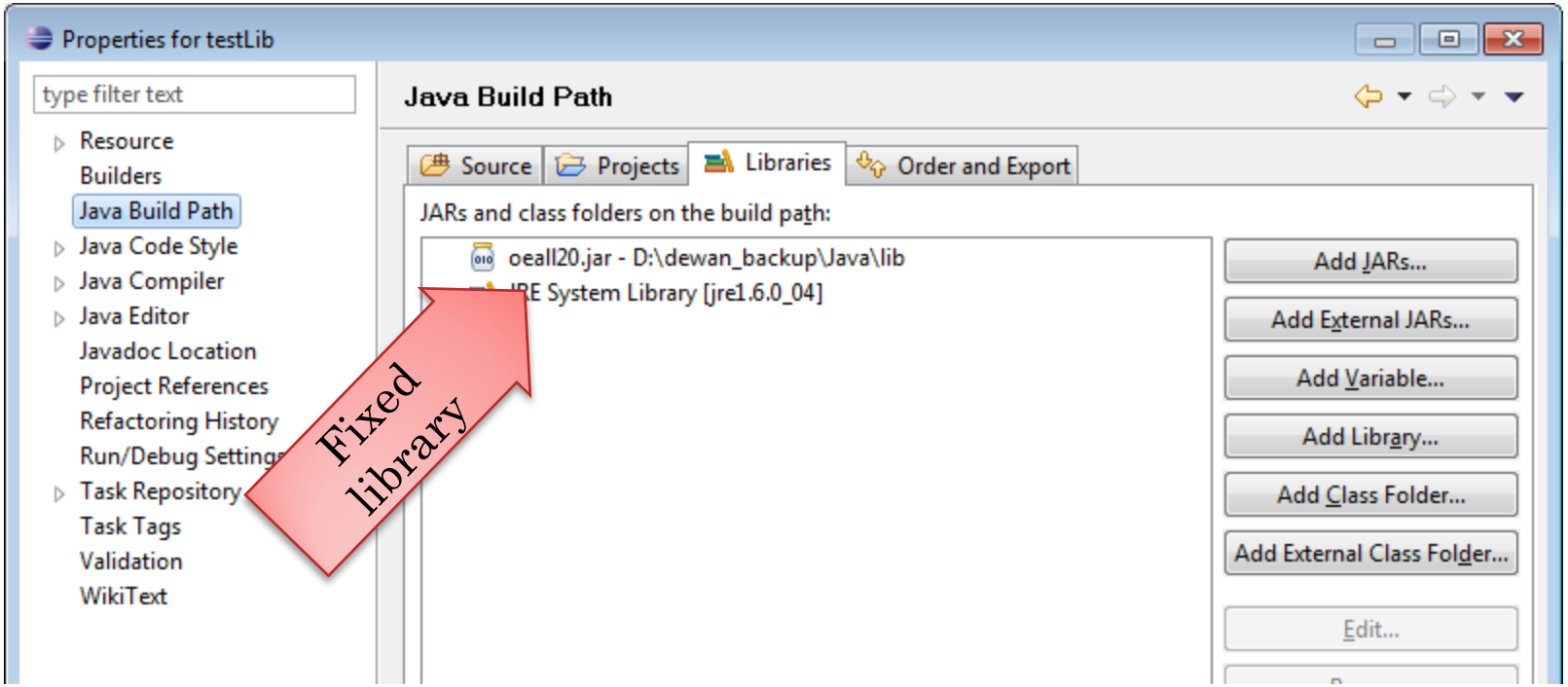
Running ObjectEditor Main

```
java –classpath .;oeall20.jar  bus.uigen.ObjectEditor
```

Library

A main class in the library

Running ObjectEditor Main

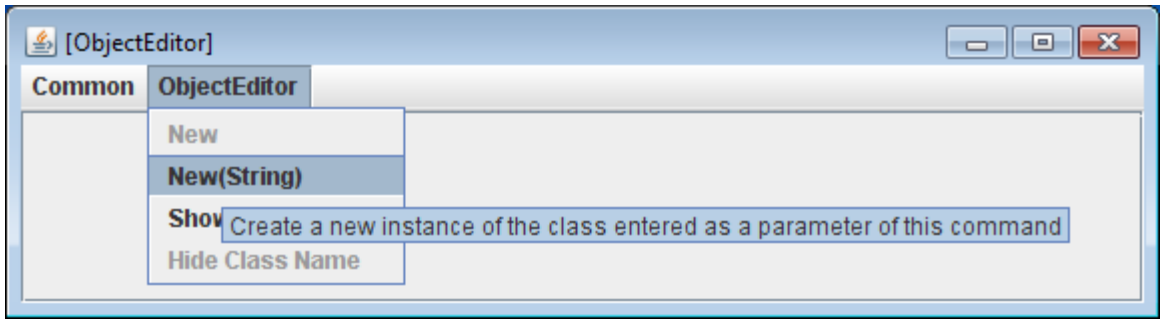


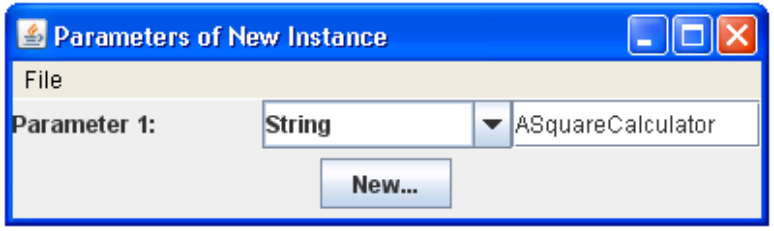
Running ObjectEditor Main

- src
- JRE System Library [jre1.6.0_04]
- Referenced Libraries
  - oeall20.jar - D:\dewan_backup\Java\lib
    - bus.uigen
      - AClassDescriptorListener.class
      - AddComponentEvent.class
      - ADynamicEnumeration.class
      - AListenableHashtableAttributeRegisterer.1.class
      - AListenableHashtableAttributeRegisterer.class
      - AListenableVectorTesterDriver.class

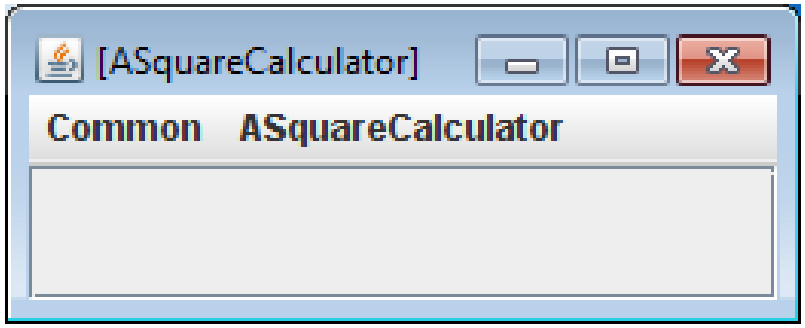
- myLockManager.class
- NodeData.class
- ObjectEditor.class
- ObjectEditorApplet.class
- ObjectEditorAR.class

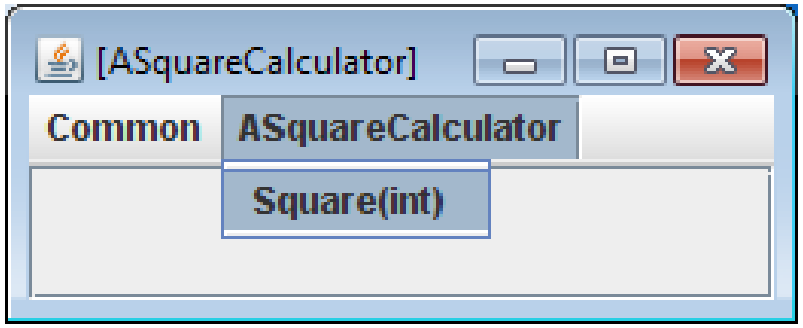
Instantiating ASquareCalculator



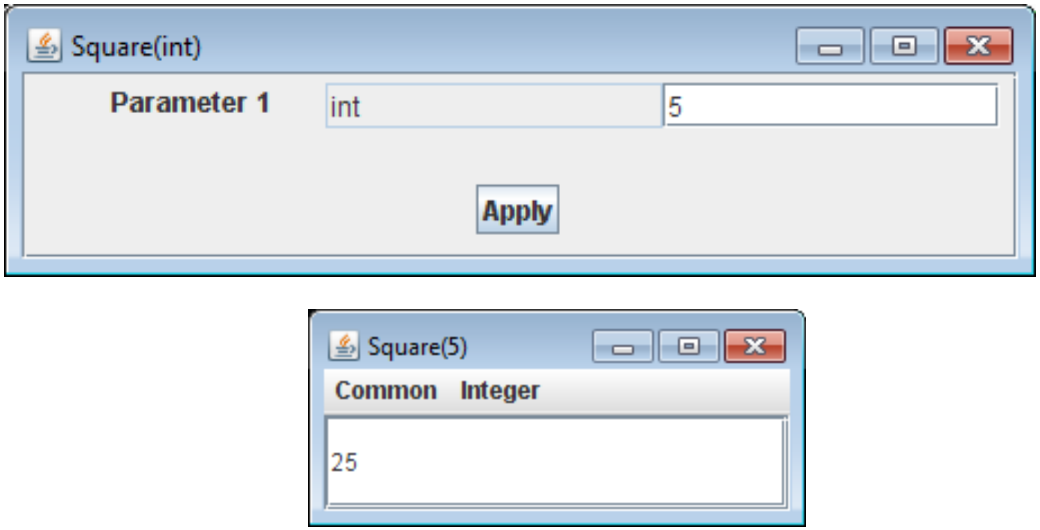


ASquareCalculator Instance

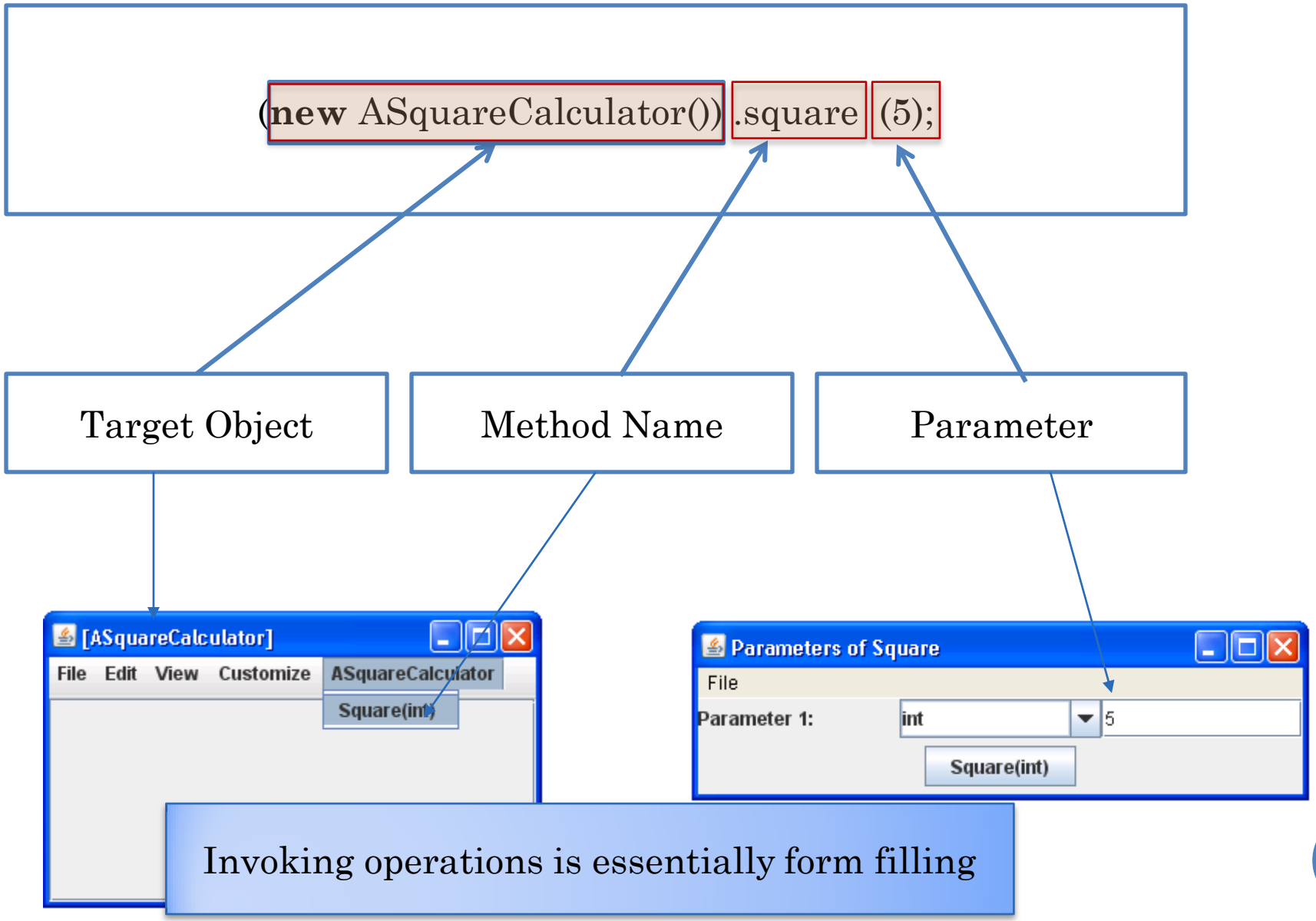




Invoking a Method And Viewing the Result



Programmatic vs. Interactive Method Invocation



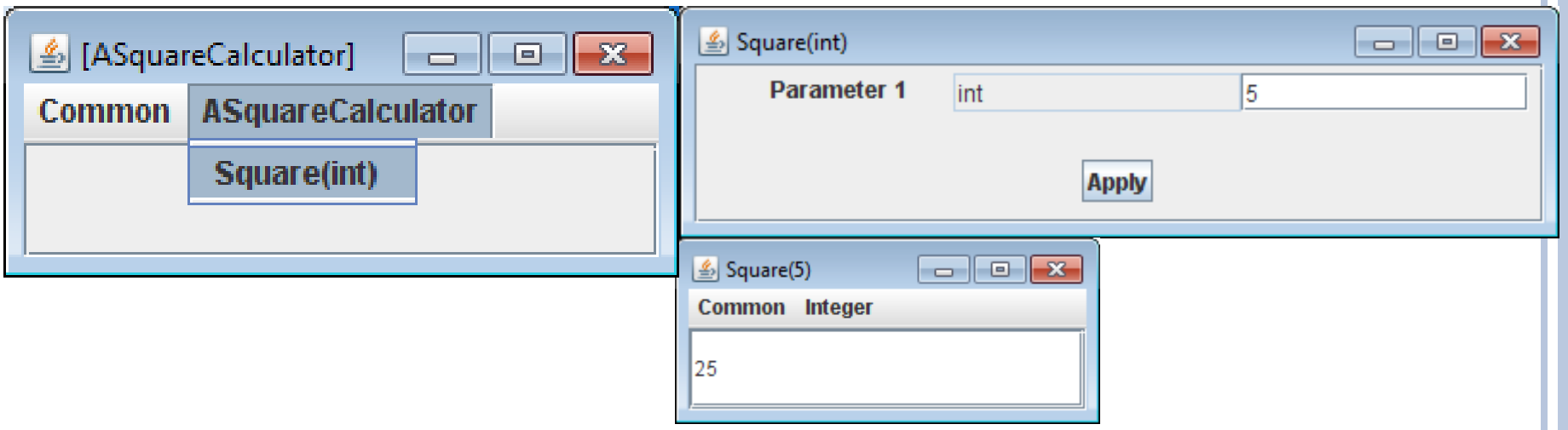
Changing Parameter

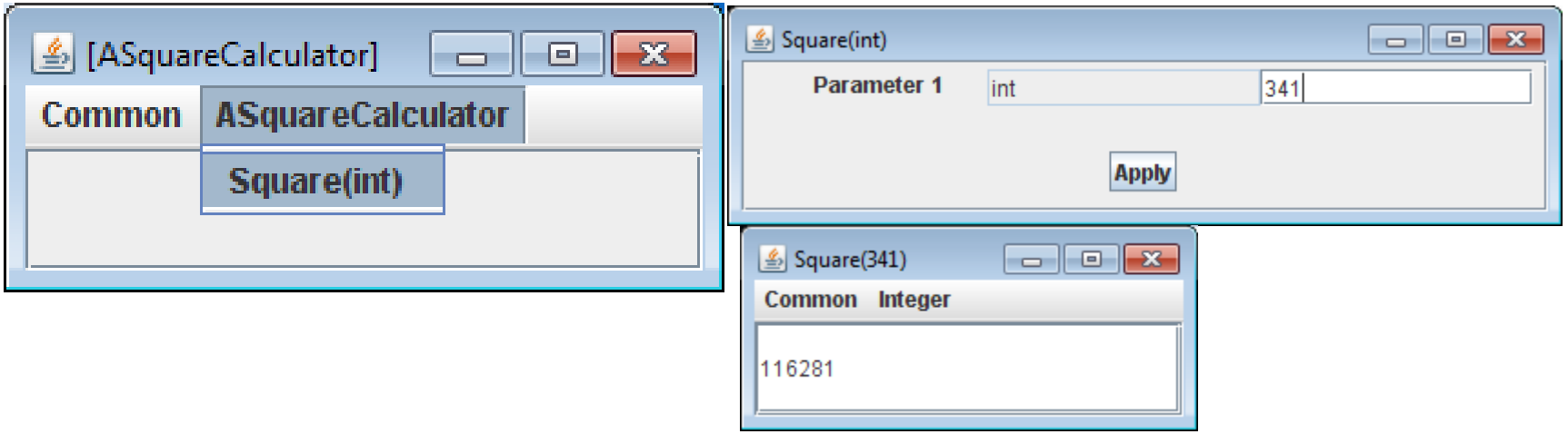
```
public class ASquareCalculator
{
    public int square(int x)
    {
        return x*x;
    }
}
```

Must change code and re-run program

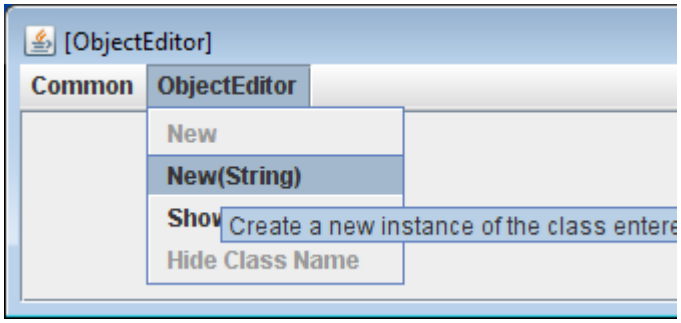
```
public class SquareCalculatorDriver
{
  public static void main (String[] args)
  {
     System.out.println (
        (new ASquareCalculator()).square(341)
     );
}
```

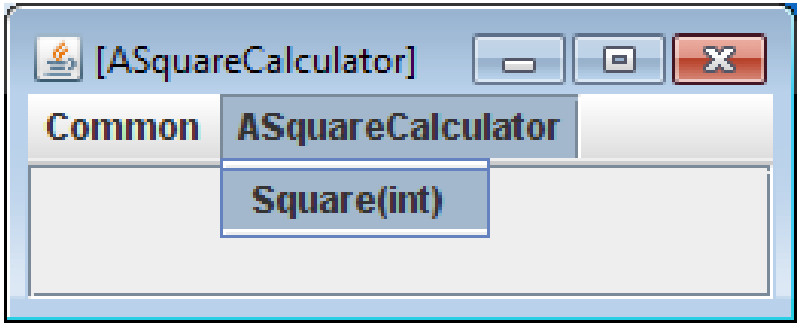
Invoking a Method And Viewing the Result





OE Capabilities





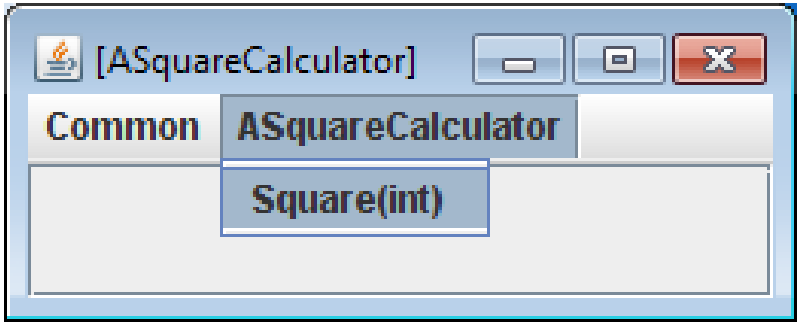
Interactive Instantiation (for beginners)

Interactive method invocation (for convenience)

Typically want to create instance in our main but invoke methods interactively

Doing Our Own Instantiation

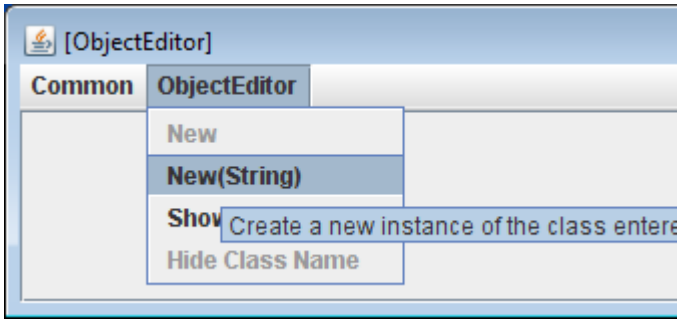
```
public class SquareCalculatorEditor {
   public static void main(String[] args) {
      bus.uigen.ObjectEditor.edit(new ABMICalculator());
    }
}
```

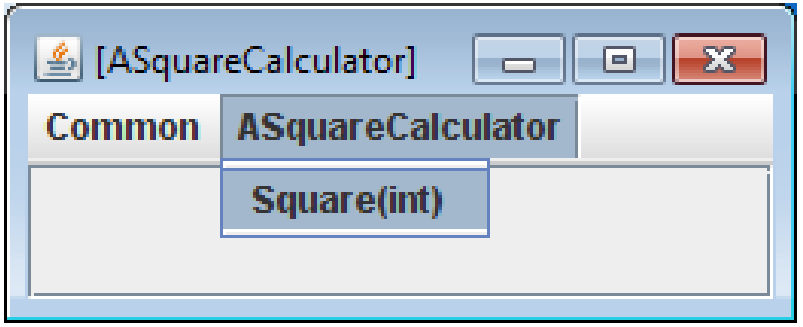


We order the car but chauffeur drives

ObjectEditor main?

ObjectEditor vs. ASquareCalculator





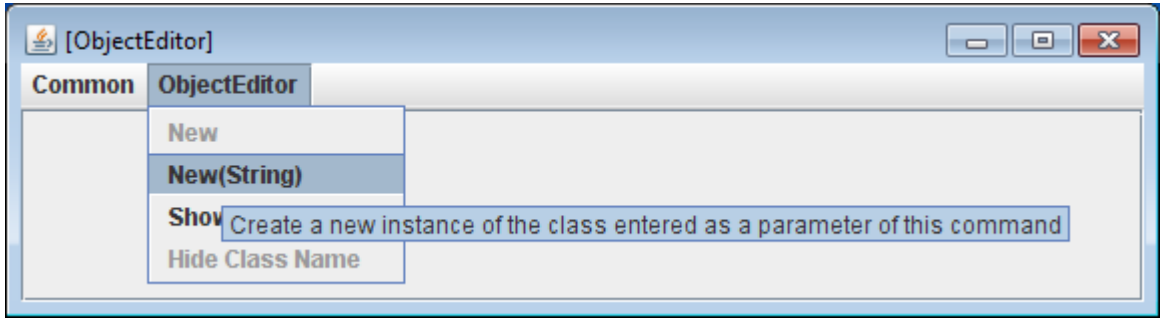
Two objects with different methods

Object editor used to edit itself

Object editor used to edit ASquareCalculator

Meta Circularity: Object Editor Main

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



Meta circularity/Boot strapping; An service provider provides itself the service

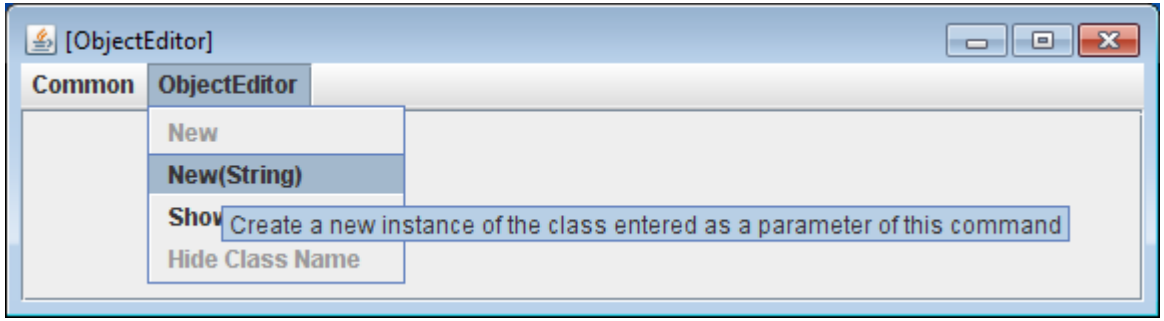
Compiler compiles itself

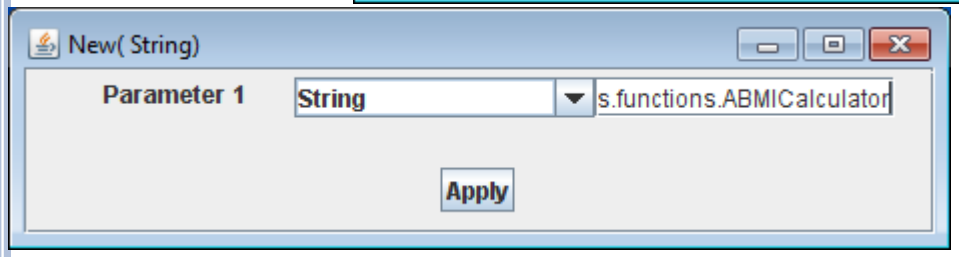
OS loads and runs itself

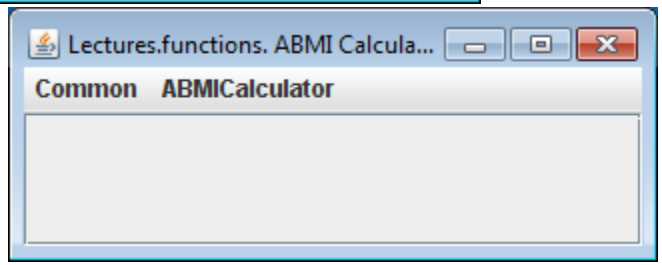
ObjectEditor edits itself

Doing Our Own Instantiation

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

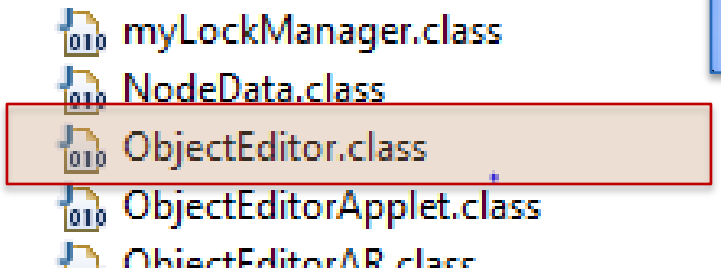






Can instantiate and edit objects without using knowing about or using main

Three Approaches



Chauffeur acquires car and drives

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

You acquire car and Chauffeur drives

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

You acquire car and drive

Integrated instantiation and method invocation

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

Instantiation and method invocation in one expression

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

Object Variables

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

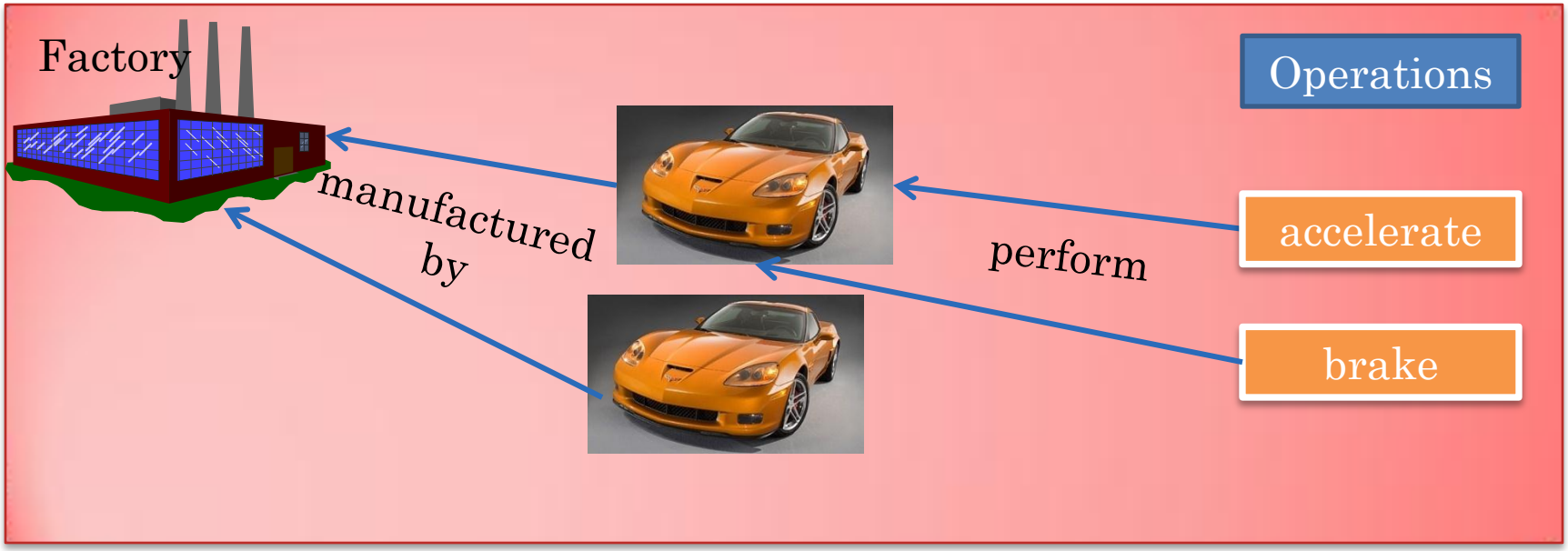
ASquareCalculator type

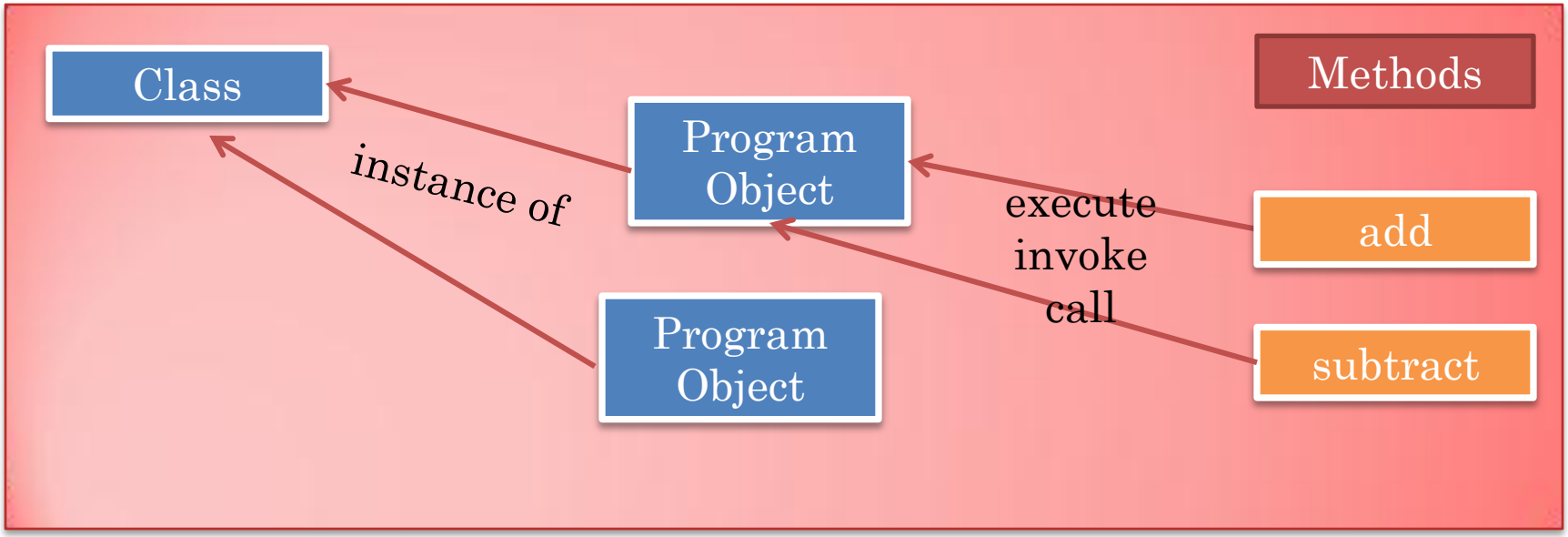
```
public class SquareCalculatorUser
{
  public static void main (String[] args)
  {
        ASquareCalculator squareCalculator = new ASquareCalculator();
        System.out.println (squareCalculator.square(5));
        System.out.println (squareCalculator.square(341));
    }
}
```

A variable that holds instances of ASquareCalculator

An instance of ASquareCalculator

Instance of


Factory
Operations
manufactured by
perform
accelerate
brake


Class
Methods
Program Object
instance of
execute
invoke
call
add
Program Object
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

ASquareCalculator

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

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

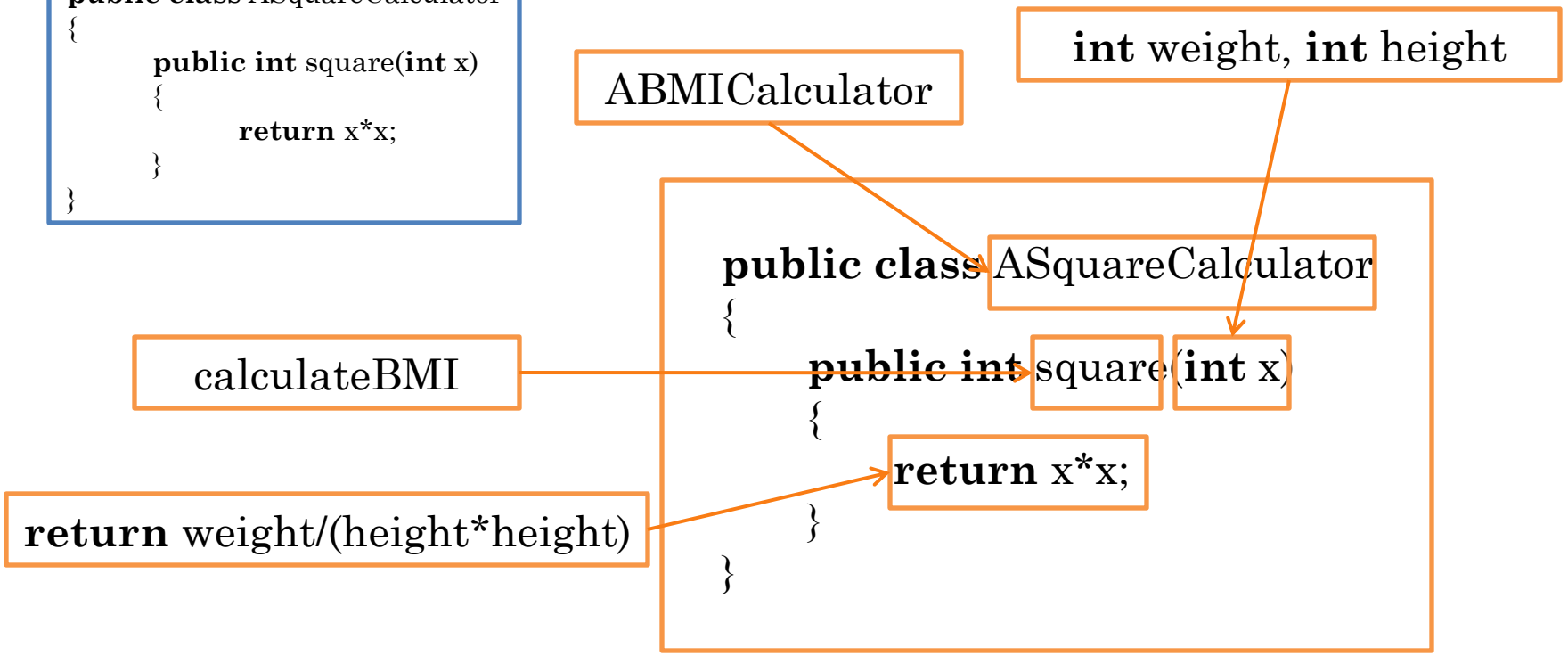
ABMICalculator

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

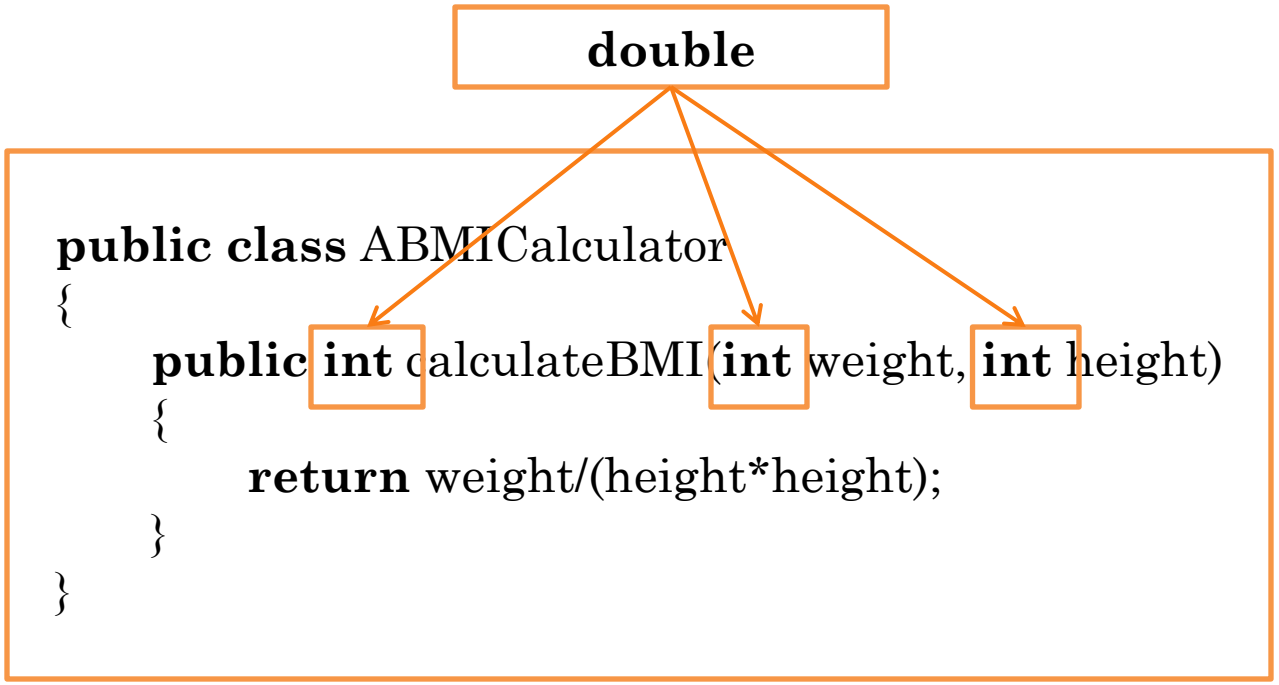
?

Multiple Parameters

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



Different type

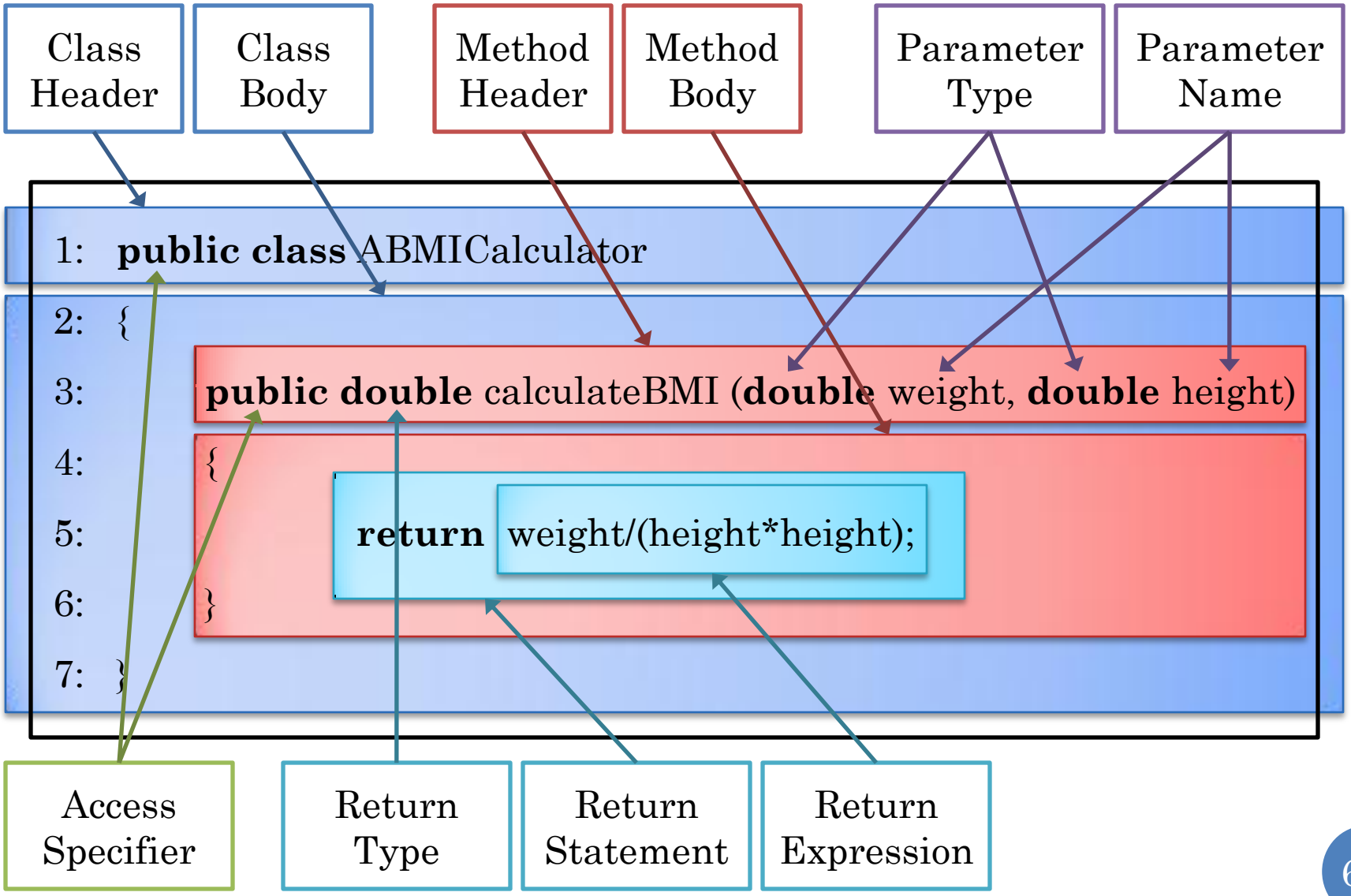


Doubles are decimal/real numbers

ABMICalculator

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

Anatomy of a Class



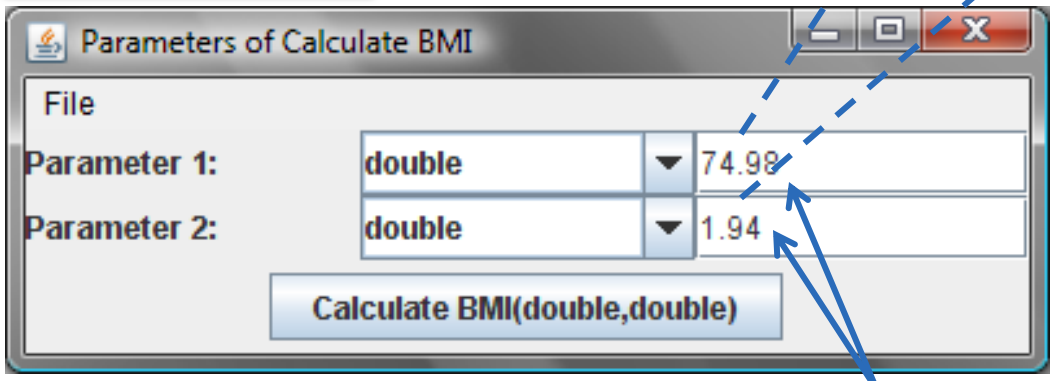
Formal vs. Actual Parameters

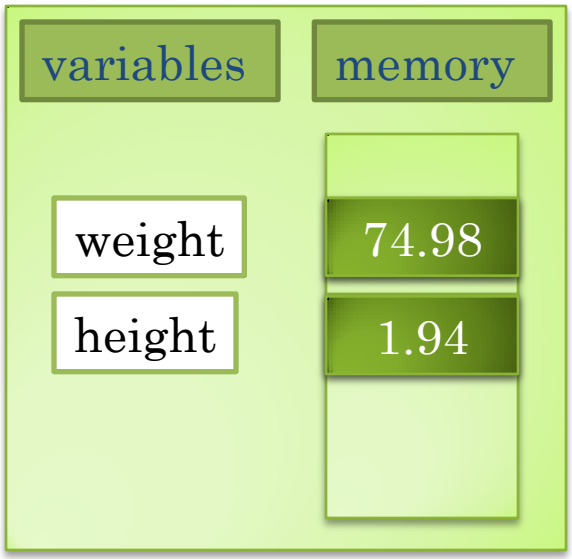
```
public class ABMICalculator
{
    public double calculateBMI(double weight, double height)
    {
        return weight/(height*height);
    }
}
```

Formal Parameters

Invoke calculateBMI

assigned





Actual Parameters

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

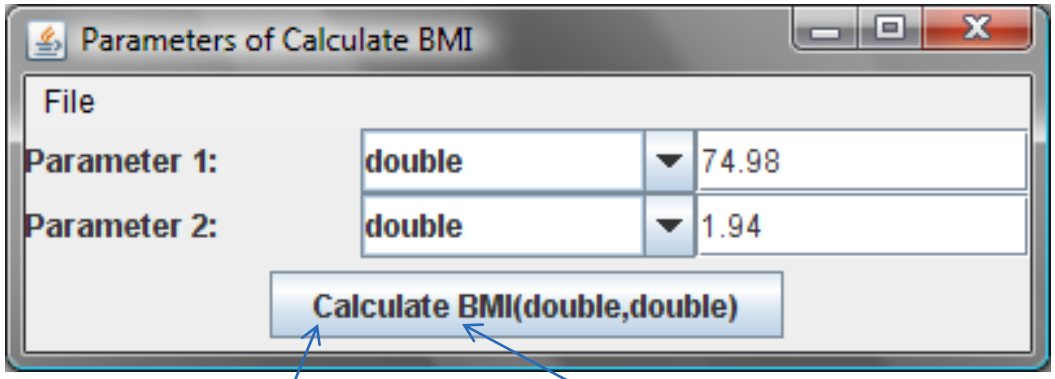
convertinches

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 int)
        {
                return x*x;
        }
}
```

Identifiers

Reserved Words (Keywords)	Program-defined Names
int, double, class, return, public	Variable names, method names, class names
boldface	Must begin with a letter
	Other characters can be letters, digits, or underscore "_"
	calculate_bmi2 calculateBMI3

Identifier Name Choices

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

Use mnemonic identifier names!

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

Mnemonic?

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

Acronyms are bad

```
public class ABMIC
{
    public double cBMI(double w, double
    {
        return w/(h*h);
    }
}
```

Use self explanatory identifier names

Self Explanatory Program?

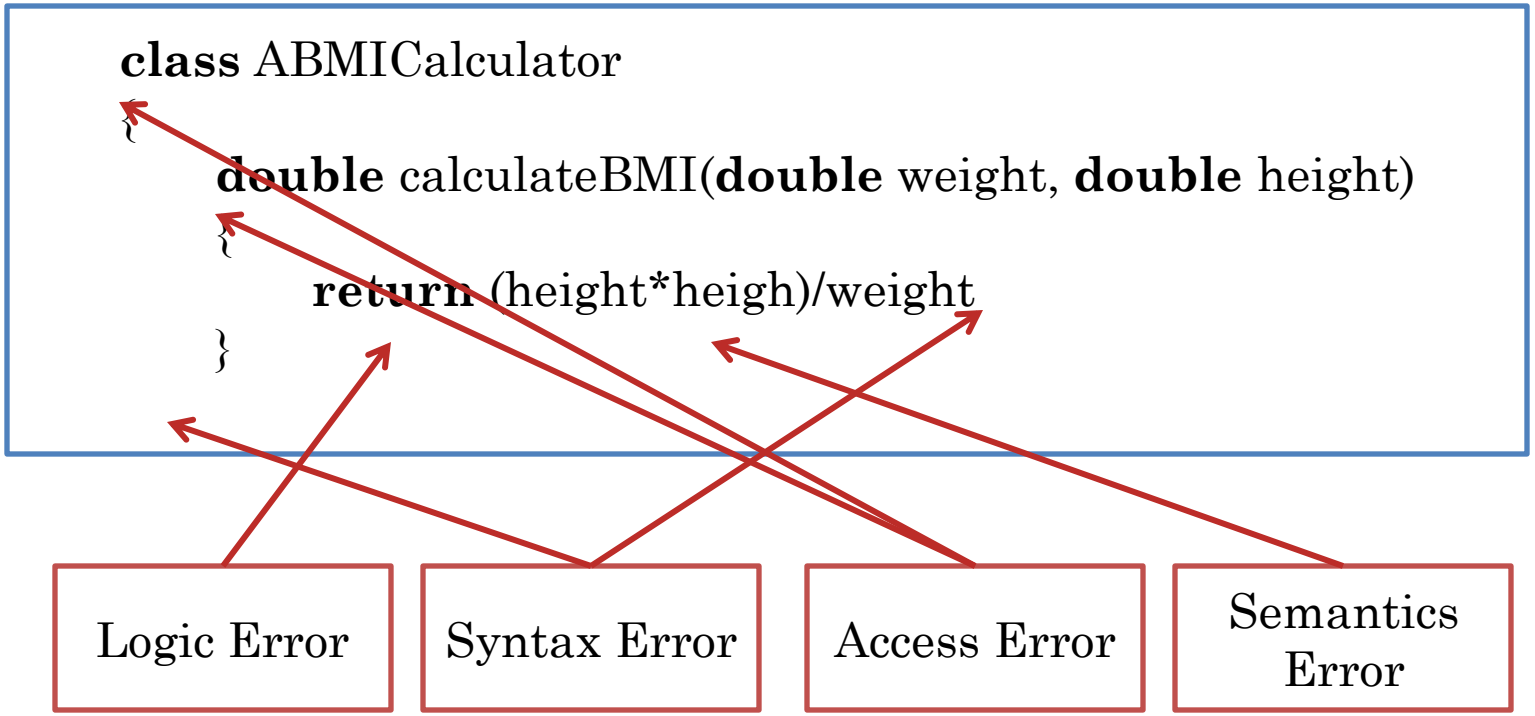
```
public class ABMICalculator
{
    public double calculateBMI(double weight, double height)
    {
        return weight/(height*height);
    }
}
```

```
public class ABMICalculator
{
    // weight is in Kgs, height in metres
    public double calculateBMI(double weight, double height)
    {
    }
}
```

Comment: Any code in the program removed by the compiler.

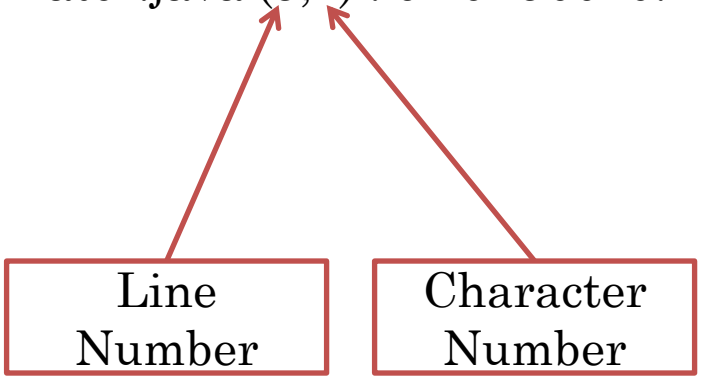
Single-line Comment: Begins with // and ends at line end.

Errors


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

Eclipse: We've Come a Long Way

Place (Hover) mouse on error

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

Click mouse on error

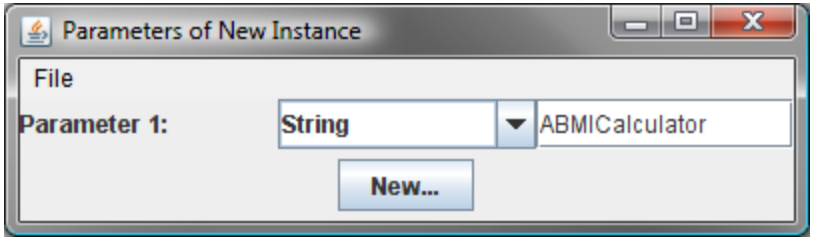
```
class ABMICalculator
{
  double calculateBMI(double weight, double height)
  {
```

Multiple markers at this line
- heigh cannot be resolved
- Syntax error, insert ";" to complete BlockStatements

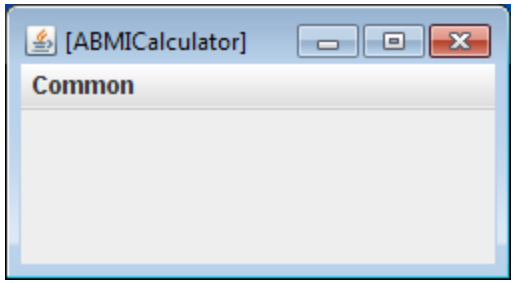
- Create local variable 'heigh'
- Create field 'heigh'
- Change to 'height'
- Create parameter 'heigh'
- Create constant 'heigh'
- Change to 'weight'
- Rename in file (Ctrl+2, R direct access)

Method Access Error

- You instantiate a ABMICalculator object



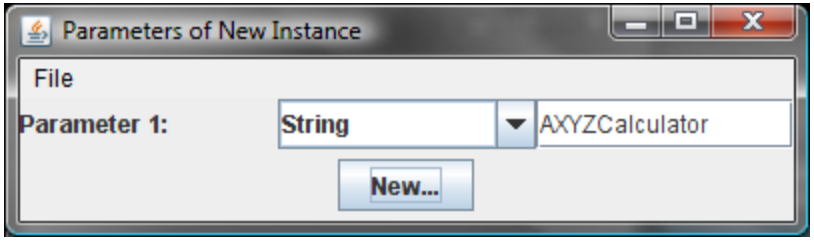
- but there is no ABMICalculator menu item



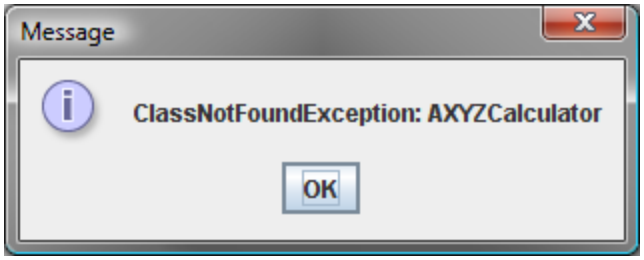
- Reason
  - You have not defined any public methods in ABMICalculator

User Error

- While instantiating a AXYZCalculator object



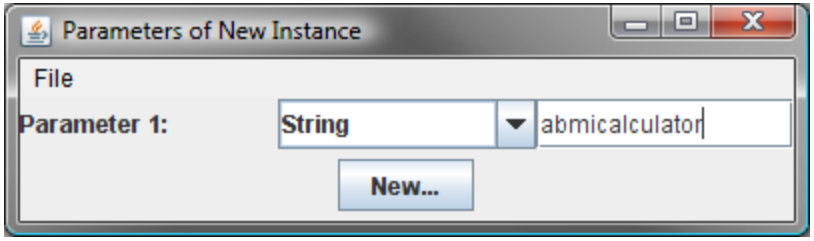
- you get the following error



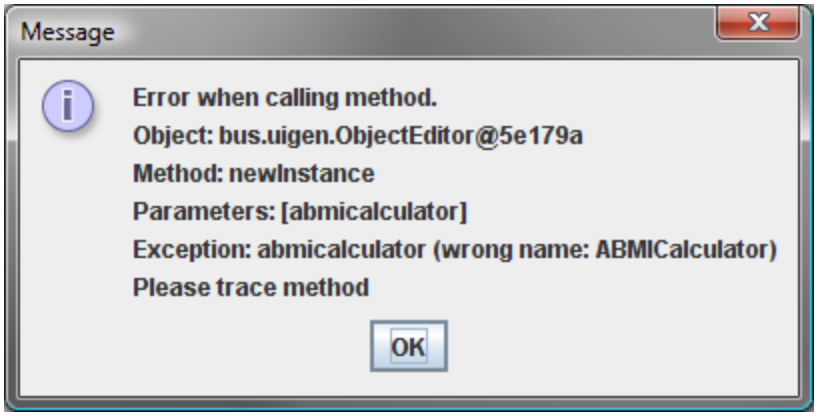
- Reason
  - AXYZCalculator class does not exist

User Error

- While instantiating a abmicalculator object



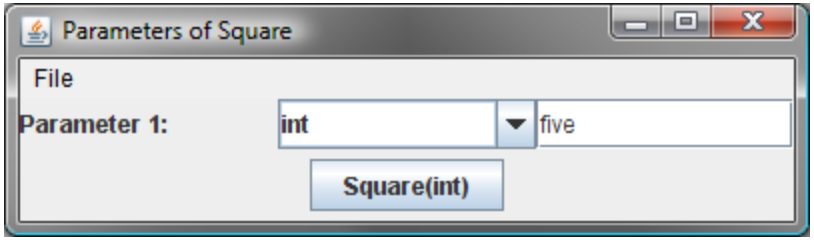
- you get the following error



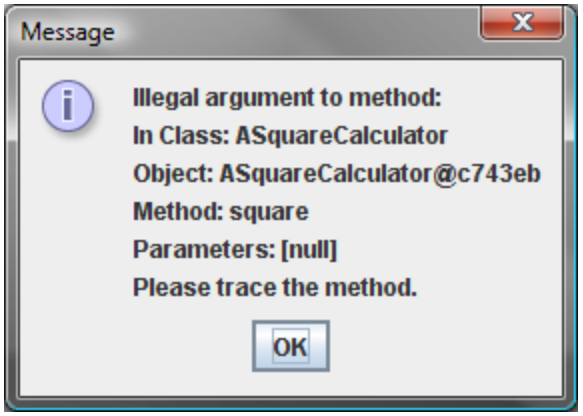
- Reason
  - Class name is spelled with incorrect case

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

Java	Real-world
Class	Factory
Computer Object	Manufactured Physical Object
Method	Operation
Invoking/Executing a Method	Performing an Operation
Instance of a Class	Manufactured by a Factory
Defining/Declaring a Class	Constructing a Factory
Instantiating a Class	Manufacturing an Object
Class (Static) Method	Operation on a Factory
Main Class Method	Initiates computation
Instance (Non static) method	Operation on an instance of a class
Instance (Non static) method	Grouping of factories by states, country

Extra slides

Changing Parameter

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

Calculates 5*5

```
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?

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

Must re-run program

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

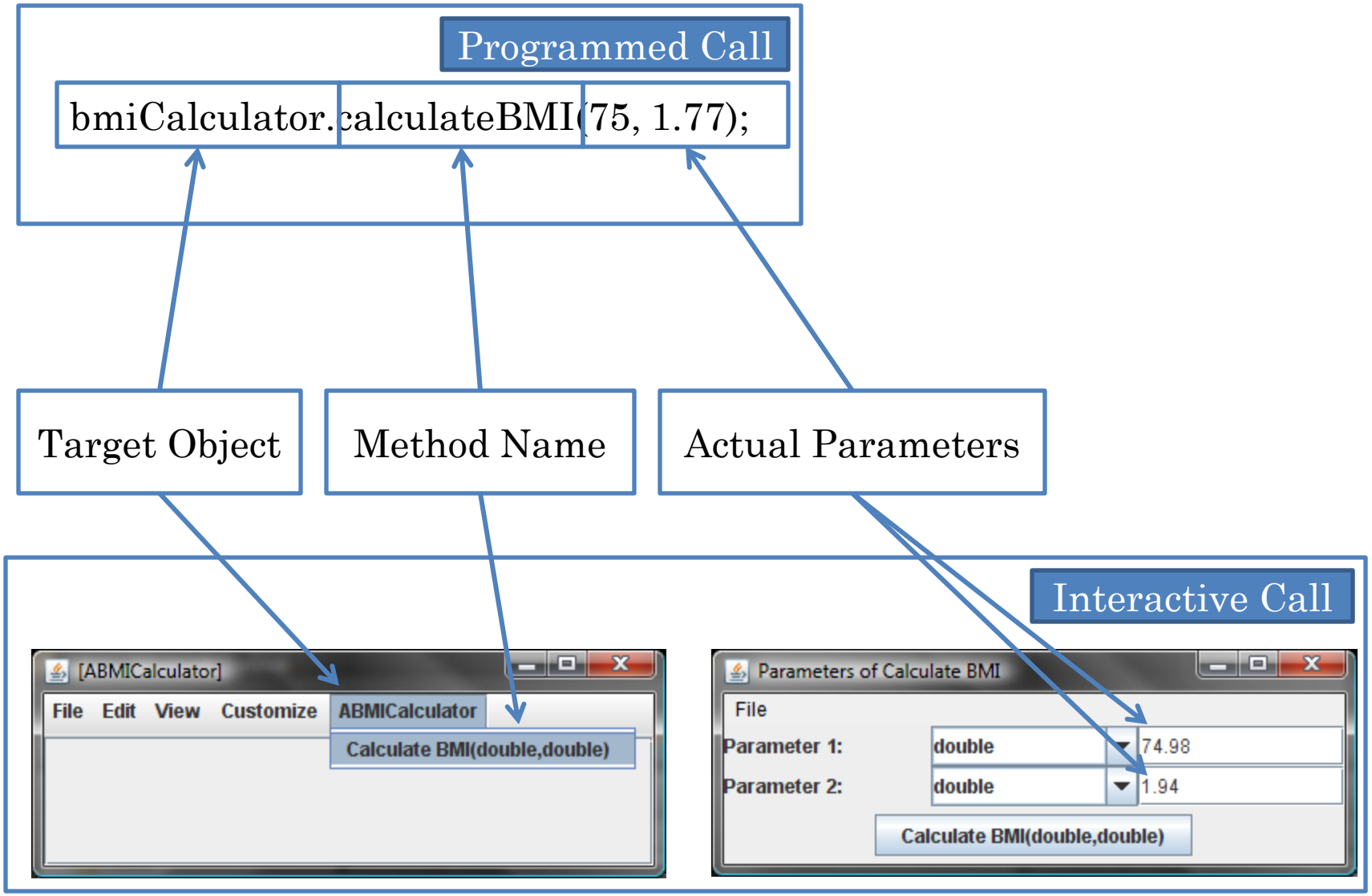
Changing Parameter

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

Must change code

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

Programmed vs. interactive call



Changing Parameter

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

Calculates 5*5

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

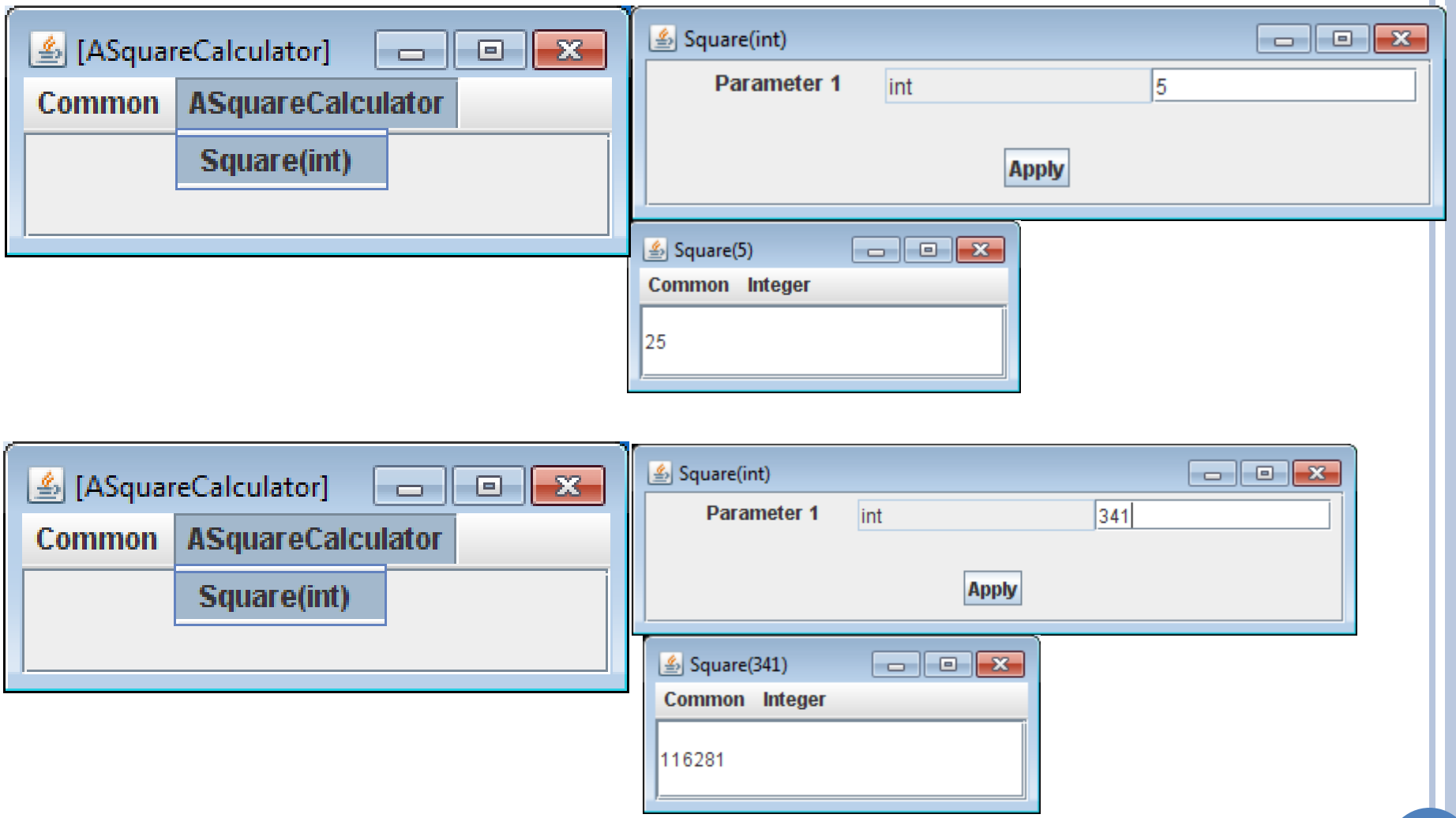
Changing Parameter

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

Must change code and re-run program

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

Invoking a Method And Viewing the Result



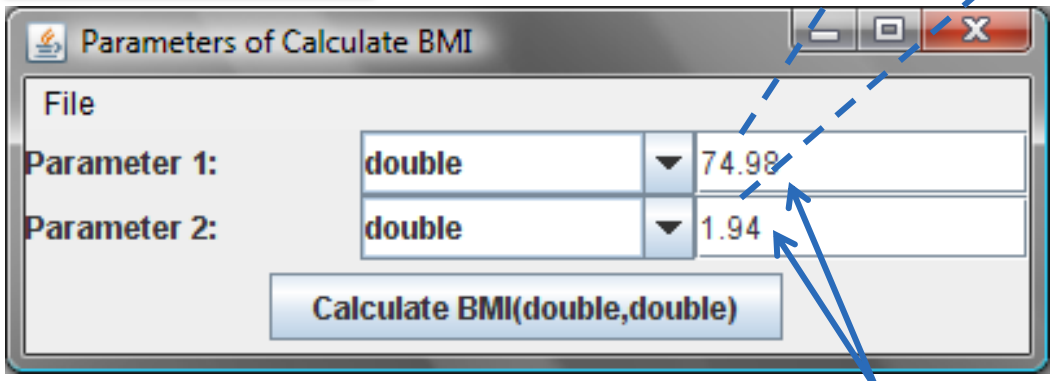
Formal vs. Actual Parameters

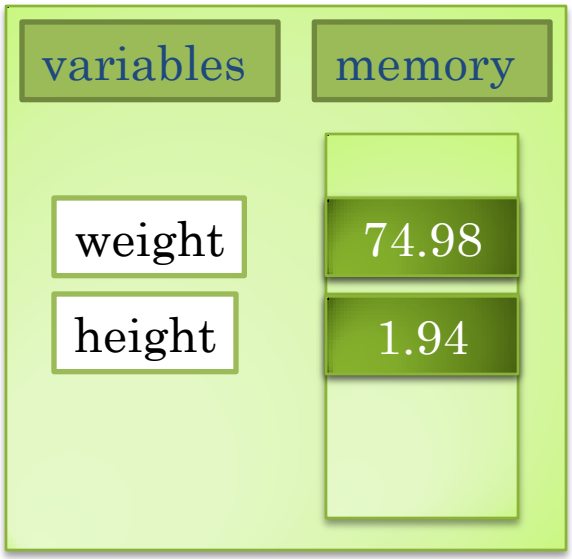
```
public class ABMICalculator {
    public double calculateBMI(double weight, double height)
    {
        return weight/(height*height);
    }
}
```

Formal Parameters

Invoke calculateBMI

assigned

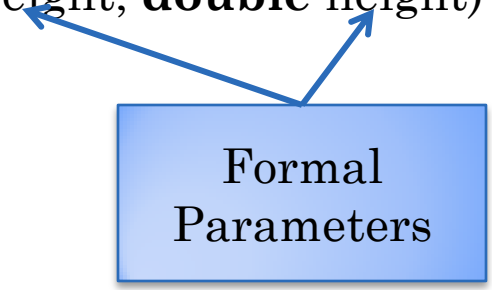




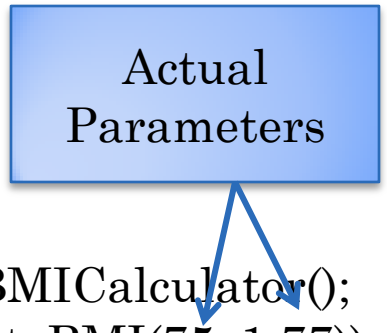
Actual Parameters

Programmed Call

```
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.out.println (bmiCalculator.calculateBMI(75, 1.77));
   }
}
```



Instance vs. Class (Static) Method

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

Instance Method

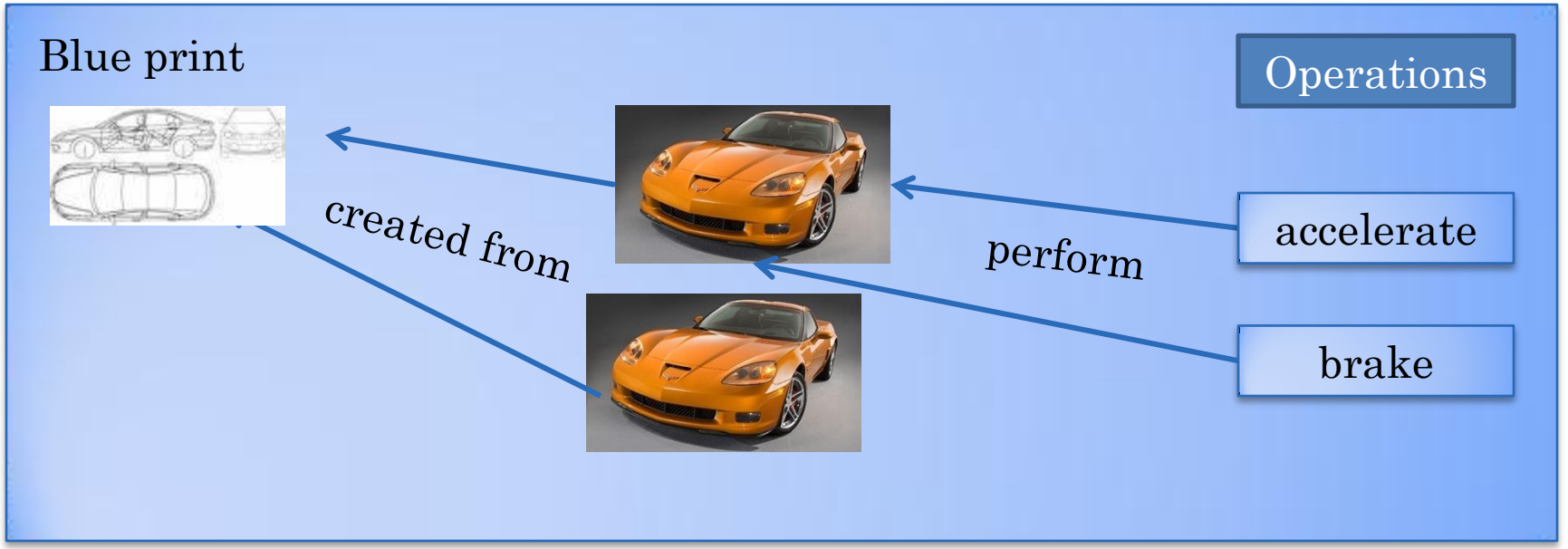
Invoked on an instance of the class

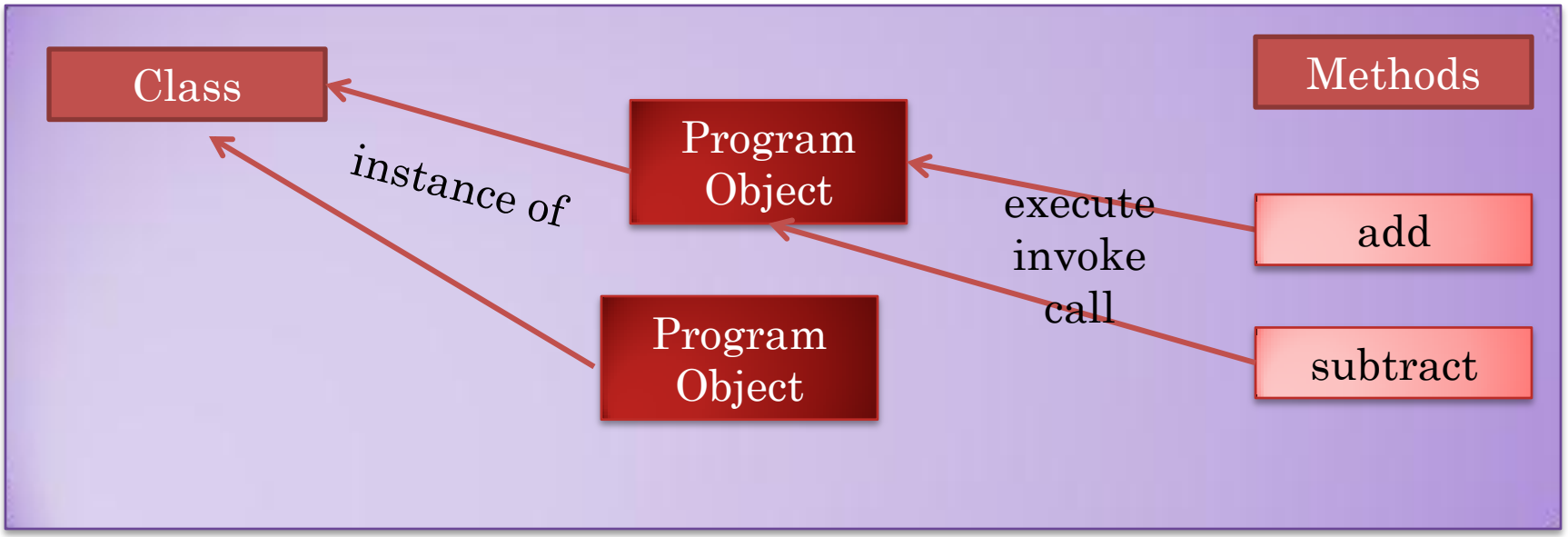
Class method

Invoked on the class

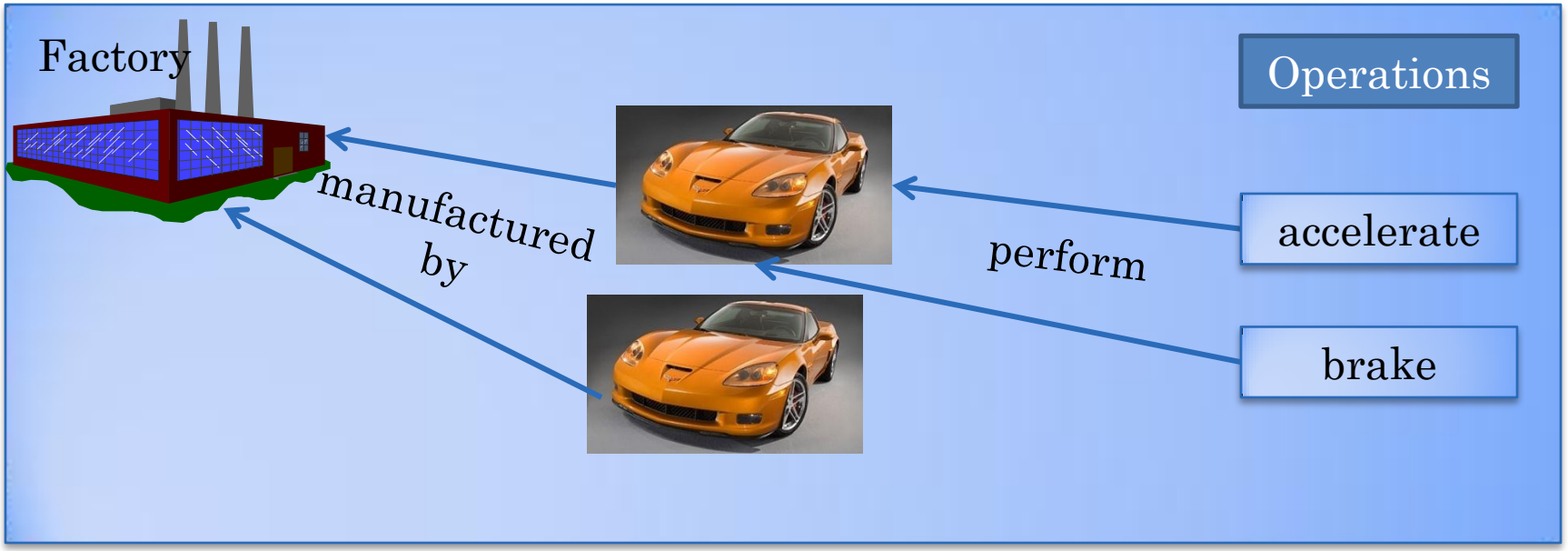
```
public class BMICalculatorDriver
{
   public static void main (String[] args) {
        ABMICalculator bmiCalculator = new ABMICalculator();
        System.out.println (bmiCalculator.calculateBMI(75, 1.77));
   }
}
```

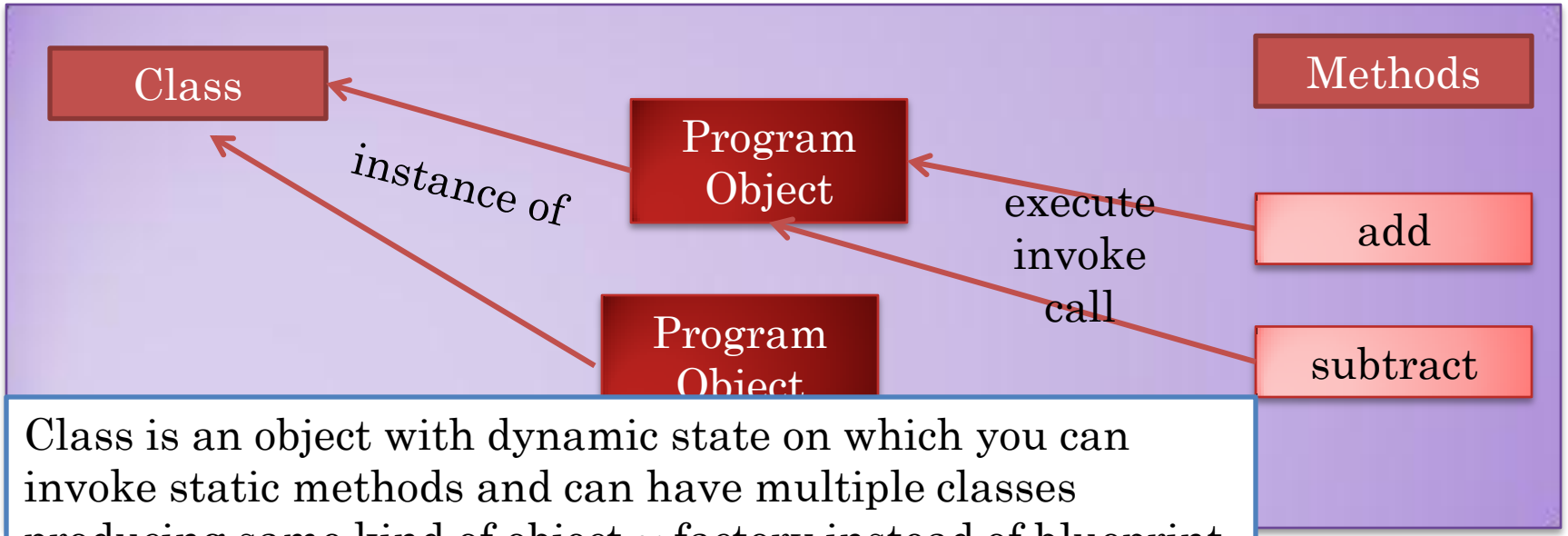
Program Objects ~ Manufactured Objects





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

ABMICalculator

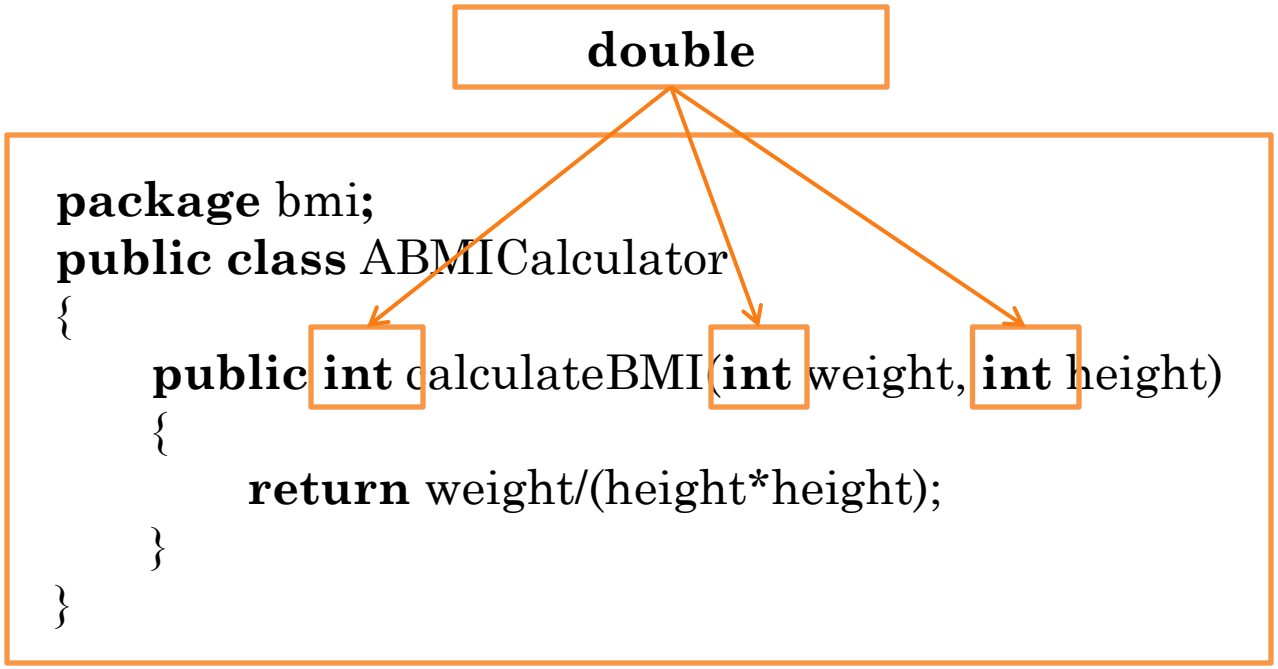
```
package bmi;
public class ABMICalculator
{
	public int calculateBMI(int weight, int height)
	{
		return weight/(height*height);
	}
}
```

Parameter and return types are integers

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

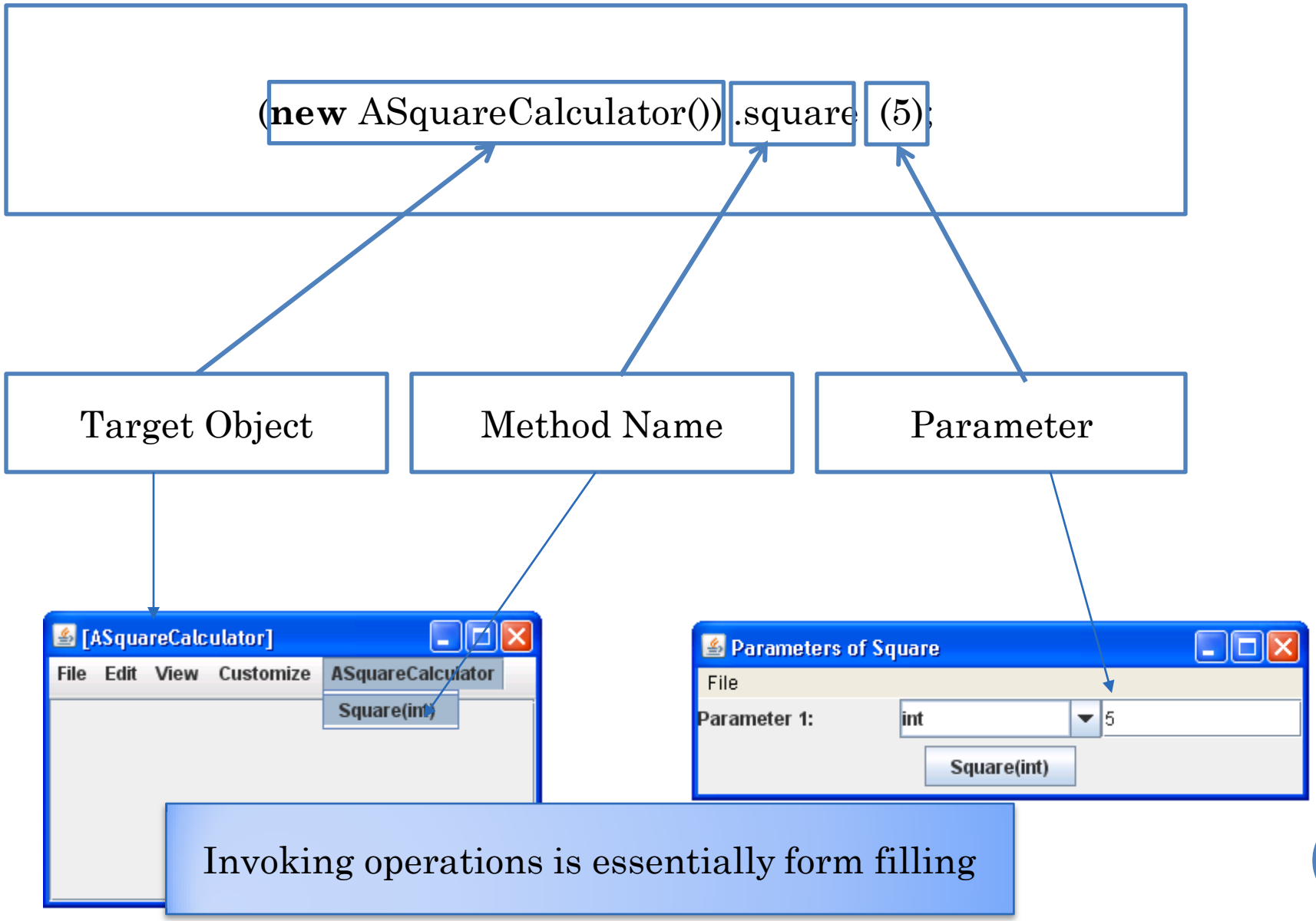
How do we solve the discrepancy?

ABMICalculator



Doubles are decimal/real numbers

Programmatic vs. Interactive Method Invocation



Object Variables

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

Operation invocation

Object Creation

Magic for beginners

```
public class SquareCalculatorUser
{
  public static void main (String[] args)
  {
      ASquareCalculator squareCalculator = new ASquareCalculator();

      System.out.println (squareCalculator.square(5));
    }
}
```