You’ve already defined
classes in lab

Syntax:

<modifiers> class <class name>
{
    <declarations>
    <methods>
}

A First Java Program

/*
Program MyFirstApplication
This program displays a window on the screen.
*/
import javabook.*;

class MyFirstApplication
{
    public static void main(String[] args)
    {
        MainWindow mainWindow;
        mainWindow = new MainWindow();
        // setVisible makes the window visible
        mainWindow.setVisible(true);
    }
}

Access

Most classes provide three levels of access to their members (state and behavior):
- Public: the part of the class of the class that is visible to all clients of the class
- Protected: the part of the class that is only visible to subclasses of the class
- Private: a part of the class that is not visible to any other classes

Default Access

- If no access modifiers are present in the declaration of a class or method, it has default access.
- Default access declares that the item may be freely accessible within a package, but not from outside of it
- This is different from public, private, and protected access!!!

You’ve already defined
methods in lab

Syntax:

<modifiers> <return type> <method name> (<parameters> )
{
    <statements>
}

A First Java Program

/*
Program MyFirstApplication
This program displays a window on the screen.
*/
import javabook.*;

class MyFirstApplication
{
    public static void main(String[] args)
    {
        MainWindow mainWindow;
        mainWindow = new MainWindow();
        // setVisible makes the window visible
        mainWindow.setVisible(true);
    }
}
Class Methods

- Constructor
- Destructor
- Behavior Methods

Return Types

- A return type from a method may be:
  - A primitive type
  - An object
  - If there is not return type, then the return type is declared as `void`

Constructors

- Syntax:
  ```java
  public <class name> ( <parameters> )
  {
    <statements>
  }
  ```

  How constructors differ from other methods

  - A constructor is a special method executed when the `new` message is sent to the class
  - Multiple constructors can be defined for a class
  - Differences:
    - No return type
    - It's not declared static, but it's a class method
    - If you do not write a constructor for a class, Java will create a dummy default constructor

Returns

```java
- return <expression>;
```

- public double returnARandomNumber()
  ```java
  {
    return Math.random();
  }
  ```

When void is used

- When there is no return value, then the word “void” is used where the method return type should be
- Void is not used when there are no parameters

Examples:
- public void run()
- public void stop()
- public void removeActor(int i)
### Static means One

- If you want a method or value to be a class method/value, then it must be declared `static`
  - `static` means that there will be exactly one incarnation of the value/method, no matter how many instances are created from a class containing the static value/method

### Examples

- Pretty much anything in the Math class!

### Final means Constant

- Sometimes these methods/values are also declared `final`, in which case their value or function is constant

```java
final double PI = 3.14;
```

### Why not public instance variables?

- See Wu book p. 158-161

### When to keep things private and why

- Declare class and instance variables private
- Declare class and instance methods private if they are used only by other methods in the same class
- Declare class and instance methods public if they are used by outside methods

### Example

- `<a Class written in class>`
If Statement

Syntax:

```
if (booleanExpression)  statement
or
if (booleanExpression)
  statement
else
  statement
```

Example

```
import java.util.*;
public class Morning {
  public static void main( String args[] ) {
    Calendar rightNow= Calendar.getInstance();
    if (rightNow.get(Calendar.AM_PM) == Calendar.AM )
      System.out.println( "Good morning..." );
    else
      System.out.println("Good afternoon..." );
  }
}
```

It doesn’t matter…

- Since
  - semicolons mark the end of an expression
  - Curly brackets mark a compound statement

- It doesn’t matter to the compiler if you put parts of a statement on different lines!

…but it does

- It matters very much to humans how your code is formatted

Dangling Else

```
if ( booleanExpression )
  if ( booleanExpression )
    statement
  else
    statement
```

- The else clause is always associated with the nearest if
- Use { ... } to change the association