An object-oriented program uses objects. To use an object in a program, we must first declare and create an object, and then we send messages to it.

**Object declarations**

Syntax: `<class name> <object names separated by commas> ;`

1. Creates a space in memory for an object, but does not create the object
2. Identifier/name: a letter followed by one or more letters/digits/underlines (no spaces allowed)

Example: `OutputBox outputBox1;`

Naming convention: classes start with an upper case letter and objects start with a lower case letter. Following words used in the same name all start with upper case letter (ex. `MyStartWindow`).

**Object creation**

Syntax: `<object name> = new <class name> ( <arguments separated by commas> );`

Example: `outputBox = new OutputBox(mainWindow);`

Can create multiple objects with the same name, but only the reference to the last object created will be saved. All the others will eventually be deallocated through garbage collection because they have no references.

**Message sending**

Syntax: `<object name> . <method name> ( <arguments separated by commas> );`

Example: `outputBox.setVisible( true );`
`outputBox.printLine("x = 5");`

Note: “sending a message” is the same as “calling a method” because the message must have a corresponding method
**Import statements**

Java classes are grouped into packages/libraries. A package can include subpackages and thus the import statement can be nested (ex. `import java.awt.image.ColorModel`)

Syntax: `import <class name>.,<object name>;`

The import statement avoids having to use the class’s fully qualified name whenever it is used. If you use more than one class in the package it’s easy to use a wildcard in importing: `import java.awt.image.*;` Note: all java package names are in lower case letters!

**Declaring a class**

Syntax: `class <class name> { <class member declarations> }`

One of the classes in a java program must be the main class, in which we define a method called main.

**Method declaration**

Syntax: `<modifiers> <return type> <method name> (<parameters>) { <method body}>`

A program template for running java: comment, import statements, class name, method body is shown on Wu pg. 58.

**Variables**

A variable has 3 properties: a memory location to store the value, the type of data stored in the memory location, and the name (identifier) used to refer to the memory location.

Syntax: `<data type> <variables separated by commas> ;`

Examples:
```java
int I, j, k;
float pi=3.14f;
OutputBox outputBox;
```

There are two kinds of variables:

1. Variables of a **primitive type** that directly contains a representation of a value of that type. Primitive types take up a fixed amount of memory and so they can be directly stored and accessed by value.
2. Variables of a **reference type** that holds a reference to an object or the value `null` (a null reference). Objects take up a variable amount of space (depending on the object) and thus the content of a variable for an object is actually the address of the object rather than the object itself (unlike primitive types where the content of the variable is its value).

**Assignment**

Assignment statement: How a value is assigned to a variable name. This is NOT like mathematical assignment!!

Syntax: `<variable> = <expression> ;`

`<expression>` is an arithmetic expression and we say that `<expression>` is assigned to `<variable>`. As an example: if `pi = 3.14159;` we say that `3.14159` is assigned to `pi`. All variables must be declared and initialized before being used.