# Lecture 3

## A First Look at Class

References:

1. Tony Gaddis, Chapter 5, Starting out with Java: From Control Structures through Objects, 7 edition
2. Herbert Schildt, Chapter 6, The Complete Reference Java 10 edition, McGraw Hill





## Chapter Topics

- Class
- Object
- Introduction to Methods
- Passing Arguments to a Method
- Local Variables
- Returning a Value from a Method

# Class

- A class is a template
  - Used to define a new type of data
  - Variables (data) and methods defined within a class
  - Members of class
    - Variables and methods defined within a class

Blue print of an object





# Box Class

```
class Box {
  double width;
  double height;
  double depth;

  // compute and return volume
  double volume() {
    return width * height * depth;
  }
}
```

# Object

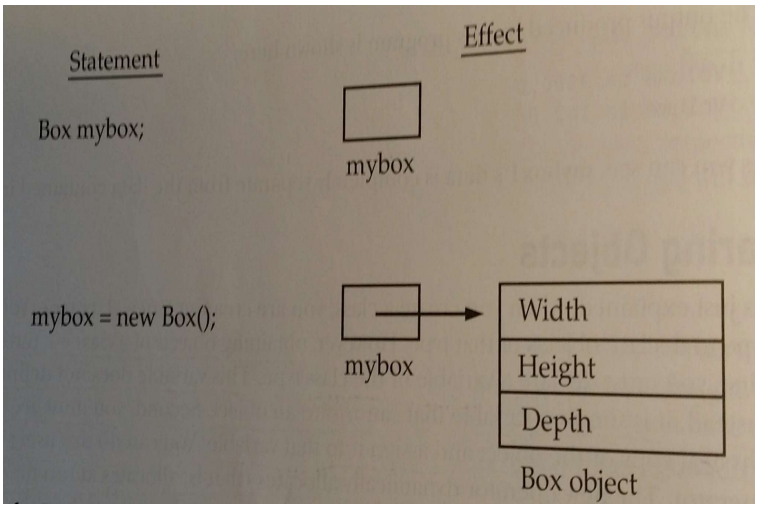
- Object of class
  - An object is an instance of a class
  - Two-step process obtaining object of a class
    - First, declare a variable of the class type
    - Second, acquire an actual physical copy of the object using "new" operator

```
Box mybox;
mybox = new Box();
or
Box mybox = new Box();
```





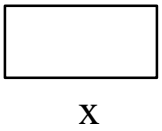
# Declaring an Object of Box class



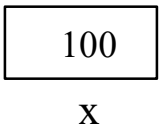
## Declaring a Variable of Primitive type

- A variable of primitive type

int x;




X = 100;








## Object

- Object of class
  - Dot(.) operator, called as a separator
    - To access both the variables and methods within an object
      - objectName.variable
      - objectName.method()

```
mybox.width
mybox.height
mybox.depth      mybox.volume()
```

  - Instance variable and method

# Object

- Object of class
  - Changes to the instance variables of one object
    - Have no effect on the instance variables of another object
  - See example: BoxDemo4.java





# new Operator

- Dynamically allocates memory for an object during run time
- Returns a *reference to the object* (address of the object)
- Cannot manipulate *a reference to an object* to point to an arbitrary memory location
- An object occupies space in memory
  - Whereas a class is a logical construct

# new Operator

- Object
  - User-defined type
  - class-var = new classname()
  - classname() is a constructor of the class

```
Box mybox = new Box();
```

```
String str = new String("Lubbock");
```

  - But String class

```
String str1 = "Lubbock";
```





# new Operator

- Primitive types
  - int, float, char, boolean, …
  - Implemented as "normal" variable
  - Do not need to use "new" operator

```
int x = 9;
```

- However, wrapper classes for primitive types

```
Integer x = new Integer(9);
Integer x = 9;
```

## Assigning Object Reference Variables

Box b1 = new Box();

Box b2 = b1;

Located at the same address

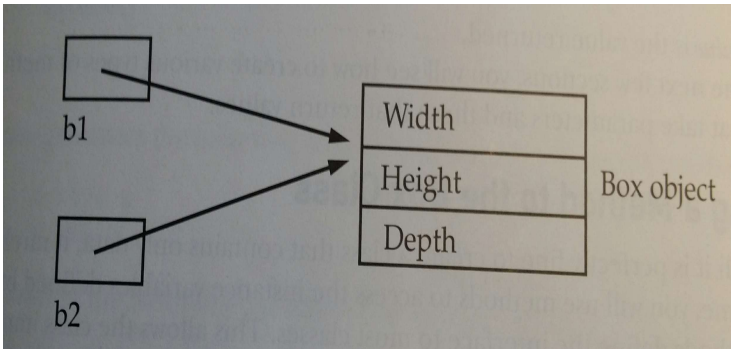
– b1 and b2 both refer to the same object

– Any change to b2 affects b1





## Assigning Object Reference Variables



## Why Write Methods?

- Methods
  - To break a problem down into small manageable pieces
- Methods reusable programs
  - Written once to perform a task, and then be executed anytime it is needed
  - Known as *code reuse*

15

## `void` Methods and Value-Returning Methods

- A `void` method
  - Simply performs a task and then terminates

```
System.out.println("Hi!");
```

- A value-returning method
  - Not only performs a task, but also sends a value back to the code that called it

```
int number = Integer.parseInt("700");
```

16

## Defining a `void` Method

- To create a method
  - Define a *header* and a *body*
- The method header
  - List several important things about the method
- The method body
  - A collection of statements

17

## Two Parts of Method Declaration

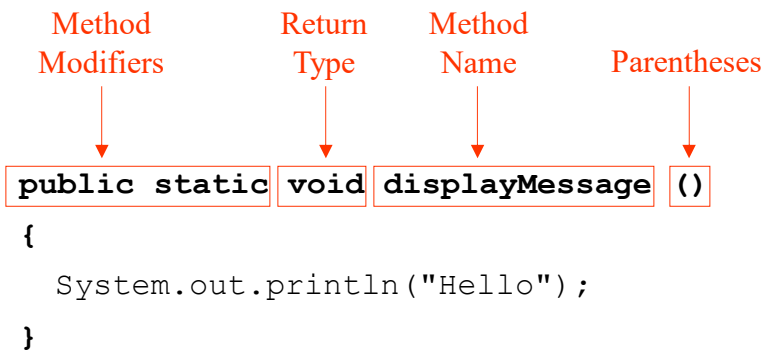
Header

```
public static void displayMesssage()
{
    System.out.println("Hello");
}
```

Body

18

## Parts of a Method Header (1 of 2)




19

## Parts of a Method Header (2 of 2)

- Method modifiers
  - `public` - publicly available to code outside the class
  - `static` - method belongs to a class, not a specific object
    - Compare to instance method
- Return type - `void` or the data type from a value-returning method
- Method name - descriptive of what the method does
- Parentheses - contain nothing or a list of one or more variable declarations

20

## Calling a Method

- The `main` method
  - Automatically called when a program starts
- Other methods
  - Executed by method call statements

```
displayMessage();
```

- Examples: SimpleMethod.java, LoopCall.java, CreditCard.java, DeepAndDeeper.java

21

## Documenting Methods

- Method Documentation
  - Writing comments that appear just before the method's definition
  - Begin with `/**` and end with `*/`

22

## Passing Arguments to a Method

- Argument
  - Values sent into a method

```
System.out.println("Hello");
number = Integer.parseInt(str);
```

- Parameter
  - The variable holding the value to be passed into a method

```
System.out.println (String x)
displayValue(int num)
```

- See example: PassArg.java

23

## Passing 5 to the `displayValue` Method

```
displayValue(5);
```

The argument 5 is copied into the parameter variable **num**.

```
public static void displayValue(int num)
{
 System.out.println("The value is " + num);
}
```

The method will display **The value is 5**

24

## Argument and Parameter Data Type Compatibility

- Argument's data type
  - Compatible with the parameter variable's data type
- Automatically perform widening conversions

```
int x = 1;
displayValue(float d);
```

- But narrowing conversions causing a compiler error

```
double d = 1.0;
displayValue(int x);
```

**Error! Can't convert `double` to `int`**

25

## Arguments are Passed by Value

- In Java, all arguments of the primitive data types
  - *Passed by value*
  - A copy of an argument's value passed into a parameter variable
  - A method's parameter variables separated and distinct from the arguments of a method
- Change of parameter inside a method
  - Has no effect on the original argument
- See example: PassByValue.java

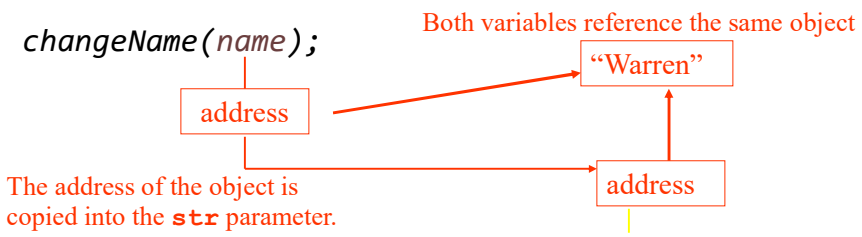
26

## Passing Object References to a Method

- A class type variable
  - Does not hold the actual data item associated with it
  - Holds the memory address of the object
  - Referred to as an (object) reference variable
- An object such as a `String` passed as an argument
  - A reference to the object passed

27

## Passing a Reference as an Argument




```
public static void changeName(String str)
{

  System.out.println("In changeName, the name " +
                     "is now " + str);
}
```

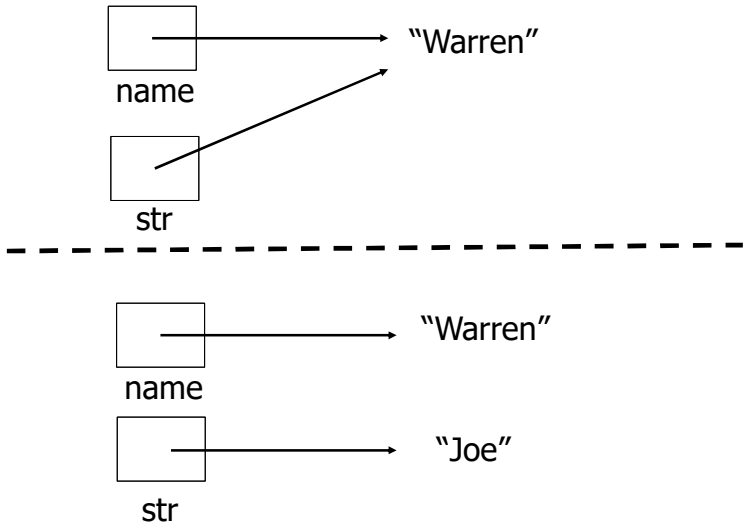
28

## `Strings` are Immutable Objects

- `String`s immutable objects
  - Cannot be changed
  - Cannot change an immutable object, so creates a new object
  - See example: PassString.jav
  - Figure

29

## PassString Program






30

## Mutable Objects

- Mutable objects
  - Can be changed
  - Can change a mutable object

```
class Test {
  int a, b;
  ……
}
```

no pointers in java

  - See example: PassObjRef.java

31

## Immutable String Object

- What for?
  - Security
    - Java class loading mechanism works on class names (string) passed as parameters
  - Thread safety
    - Immutable objects are safe when shared between multiple threads



32

## `@param` Tag in Documentation Comments

- `@param` tag
  - A description of each parameter in your documentation comments
- General format

  **`@param parameterName Description`**
- See example: TwoArgs2.java
- The description can span several lines

33

## Local Variables

- A local variable declared inside a method
  - Not accessible to statements outside the method
  - Different methods can have local variables with the same names
  - Exist only while the method is executing
- See example: LocalVars.java

34

## Returning a Value from a Method

- Data passed into a method by parameter variables
- Method maybe return a value back to the statement that called it

```
int num = Integer.parseInt("700");
```

35

## Defining a Value-Returning Method

```
public static int sum(int num1, int num2)
{
 int result;
 result = num1 + num2;
 return result;
}
```

Return type

The `return` statement causes the method to end execution and it returns a value back to the statement that called the method.

This expression must be of the same data type as the return type

36

## Calling a Value-Returning Method

```
total = sum(value1, value2);
```

20 40

```
public static int sum(int num1, int num2)
{
  int result;
  result = num1 + num2;
  return result;
}
```

60

37

## `@return` Tag in Documentation Comments

- `@return` tag
  - A description of the return value in your documentation comments
- General format

  **`@return Description`**
- See example: ValueReturn.java
- The description can span several lines

38

## Returning a Reference to a `String` Object

```
customerName = fullName("John", "Martin");

public static String fullName(String first, String last)
{
        String name;
        name = first + " " + last;
        return name;
}
```

address

"John Martin"

Local variable `name` holds the reference to the object. The return statement sends a copy of the reference back to the call statement and it is stored in `customerName`.

See example:

ReturnString.java

39

## Backup Slides



40