Lecture 5

A Third Look at Class

References:

1. Tony Gaddis, Chapter 8, Starting out with Java: From Control Structures through Objects, 7 edition
2. Herbert Schildt, Chapter 7, The Complete Reference Java 10 edition, McGraw Hill

1

1

Chapter Topics

- Static Class Members
- Object (Class) Aggregation
- The this Reference Variable
- Enumerated Types
- Garbage Collection
- Command line arguments
- Variable length arguments

2

2

Understanding static

- Normally, a class member
  - Accessed only by means of an object of its class
- When a member is declared static
  - Can be accessed before any objects of its class are created
  - Can declare both methods and variables with `static`
- E.g., `StaticByName.java` program

3

3

Understanding static

- Static variable (global variable)
  - When objects of its class are declared, no copy of a static variable is made
  - All instances of the class share the same static variable

4

4

Understanding static

- Static methods have several restrictions
  - Can only directly access static variables
  - Can only directly call other static methods
  - Cannot refer to **this** or **super** in any way

5

5

Understanding static

- Static block
  - To initialize static variables by declaring a static block
  - Executed exactly once when the class is first loaded
  - E.g., `UseStatic.java` program
  - E.g., `UseStatic2.java` program

6

6

Introducing final

- Keyword `final`
  - Prevent its contents from being modified
  - Must initialize a final field when declared
  - E.g.,

```
final int File_New = 1;
```

- Declaring a parameter final
  - To prevent it from being changed within the method
- Declaring a local variable final
  - To prevent it from being assigned a value more than once

7

7

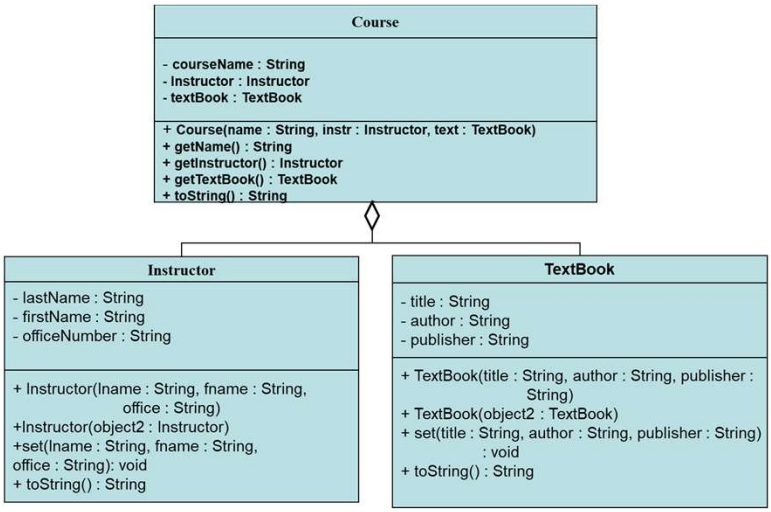
Aggregation

- *Object aggregation*
  - Creating an instance of one class as a set of another objects
  - The “Is part of” relationship between objects
- Examples:
  - Instructor.java, Textbook.java, Course.java, CourseDemo.java

8

8

Aggregation in UML Diagrams



9

9

The `this` Reference

- The `this` reference
  - An object can use to refer to itself
  - Overcome shadowing
    - Allow a parameter to have the same name as an instance field

```
public void setFeet(int feet)
{
   this.feet = feet;
   //sets the this instance's feet field
   //to the parameter feet.
}
```

Local parameter variable feet

Shadowed instance variable

10

10

Enumerated Types (1 of 2)

- `enum` type
  - A specialized class
- Syntax:

  `enum` *typeName* { *one or more enum constants* }

  - Definition:

    ```
    enum Day { SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
               THURSDAY, FRIDAY, SATURDAY }
    ```

  - Declaration:

    ```
    Day WorkDay; // creates a Day enum
    ```

  - Assignment:

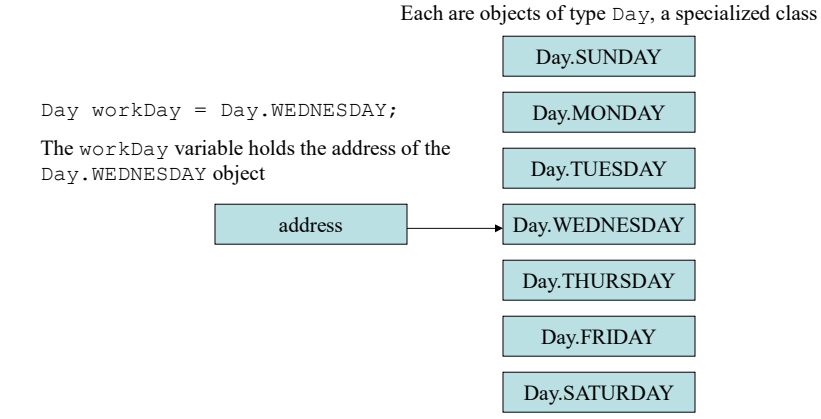
    ```
    Day WorkDay = Day.WEDNESDAY;
    ```

11

11

Enumerated Types (2 of 2)

```
enum Day { SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
THURSDAY, FRIDAY, SATURDAY }
```



12

12

Garbage Collection (1 of 5)

- The Java Virtual Machine
  - has a garbage collector process
- The *garbage collector*
  - Reclaim memory from any object that no longer has a valid reference pointing to it

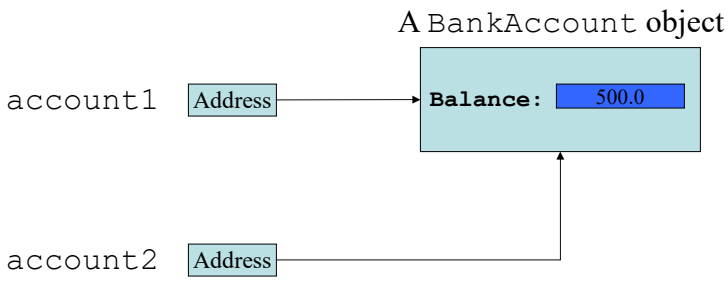
```
BankAccount account1 = new
BankAccount(500.0);
BankAccount account2 = account1;
```

- This sets `account1` and `account2` to point to the same object.

13

13

Garbage Collection (2 of 5)

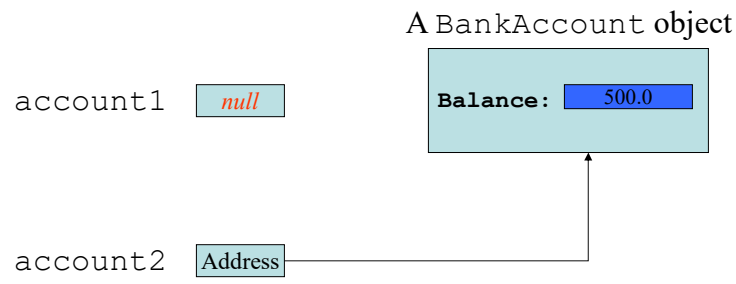


Here, both `account1` and `account2` point to the same instance of the `BankAccount` class.

14

14

Garbage Collection (3 of 5)

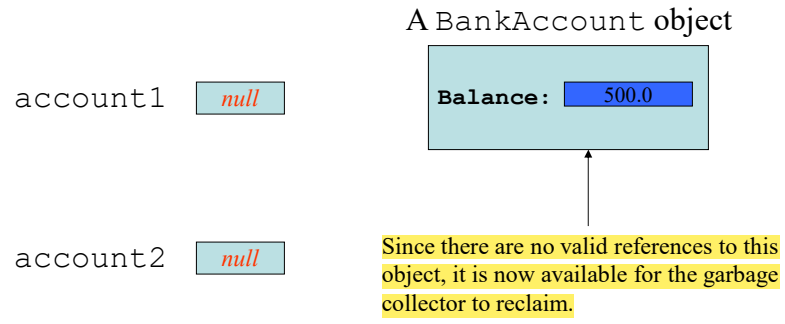


However, by running the statement: **`account1 = null;`**
only `account2` will be pointing to the object.

15

15

Garbage Collection (4 of 5)

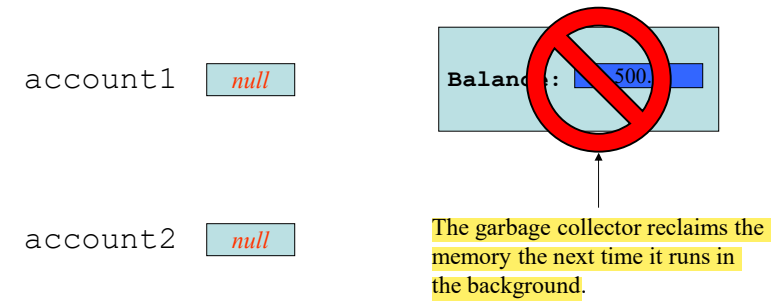


If we now run the statement: **`account2 = null;`**
neither `account1` or `account2` will be pointing to the object.

16

16

Garbage Collection (5 of 5)



17

17

Using Command-Line Arguments

- Pass information into a program when you run it
  - Command-line arguments to main()
    - Information that follows the program's name on command line
  - arg[0], arg[1], arg[2], …
  - E.g., CommandLine.java

18

18

Varargs: Variable-Length Arguments

- Varargs (variable-length arguments)
  - varargs method in Java
    - A method that takes a variable number of arguments
  - C++: Variadic functions
  - C#: param arrays (parameter arrays)

19

19

Varargs: Variable-Length Arguments

- Variable-length argument specified by three periods (…)
  - A variable-length arguments method called with zero or more arguments
  - E.g., PassArray.java
  - E.g., VarArgs.java

20

20

Varargs: Variable-Length Arguments

- A method with "normal" parameters along with a variable-length parameter
  - Must be the last parameter in a method
  - Must be only one varargs parameter in a method
  - E.g., VarArgs2.java

21

21

Varargs: Variable-Length Arguments

- Overloading Varargs methods
  - A varargs method overloaded
  - E.g., VarArgs3.java
    - vaTest(int x): one int argument is present
    - vaTest(int …v): two or more int arguments are passed

22

22

Varargs: Variable-Length Arguments

- Varargs and Ambiguity
  - E.g., VarArgs4.java
    - vaTest() – do not know which one is called
      - vaTest(Boolean … v) and vaTest(int … v)
  - Another example with vaTest(1)
    - static void vaTest(int … v) and
    - static void vaTest(int n, int … v)