Java Class 6


©Marty Bucella
www.martybucella.com
"I'm okay with the 'Man's Best Friend' thing, but I think my owner is pushing it with this Facebook friend request."

Objects and the Heap

The number of objects that a program creates is not known until the program is actually executed

Java uses a memory area called the heap to handle this situation, which is a dynamic pool of memory on which objects that are created are stored

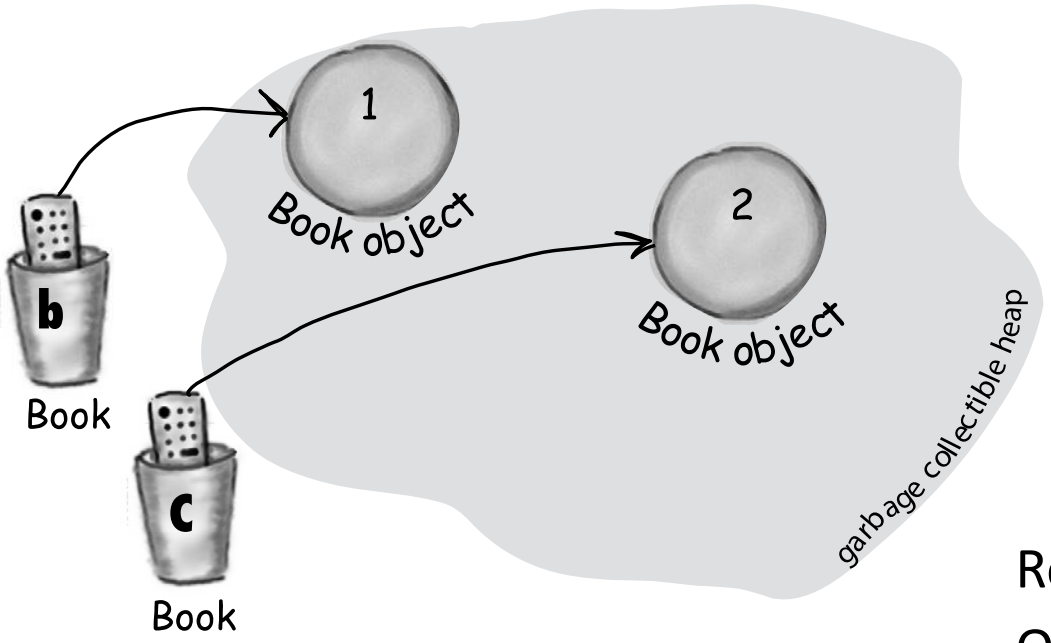
When an object is created, it is placed in the heap

Two or more references that refer to the same object are called *aliases* of each other

- One object can be accessed using multiple reference variables
- Changing an object through one reference changes it for all of its aliases, because there is really only one object

Example

```
Book b = new Book();
Book c = new Book();
```

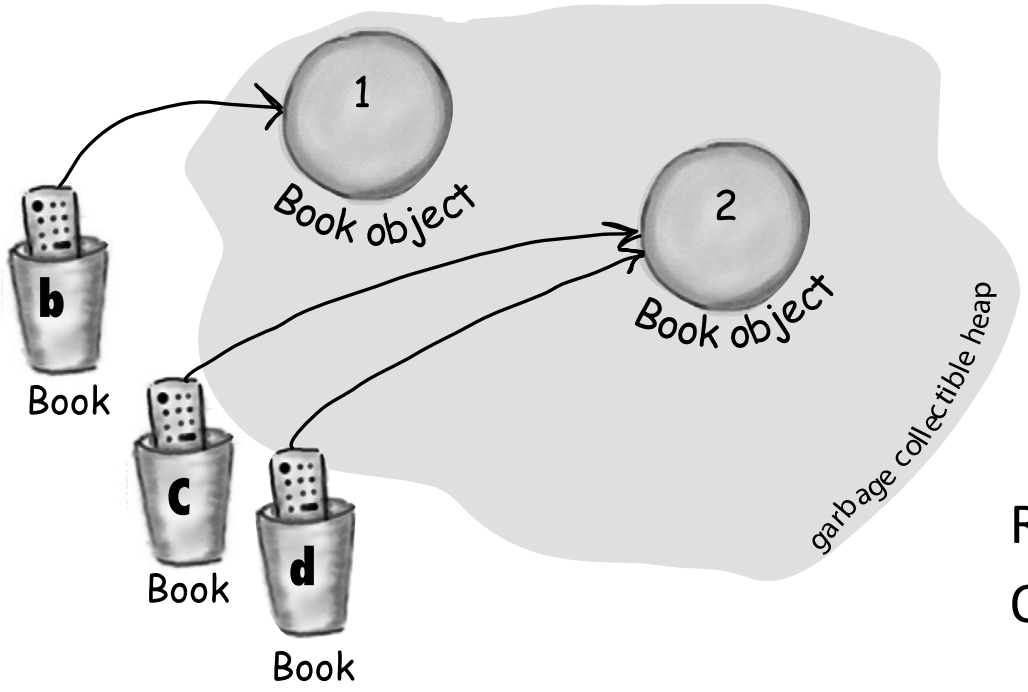


References: 2

Objects: 2

Example

```
Book b = new Book();
Book c = new Book();
Book d = c; // d and c are alias
```

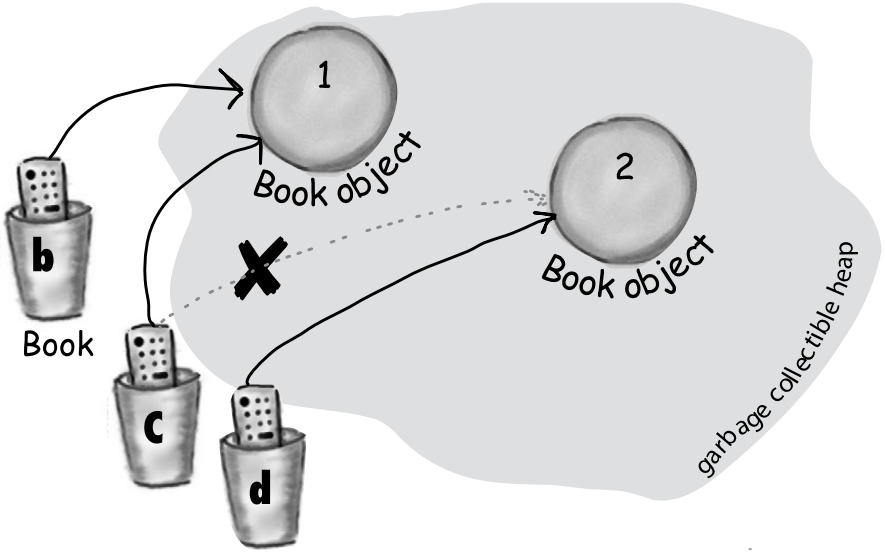


References: 3

Objects: 2

Example

```
Book b = new Book();
Book c = new Book();
Book d = c;
c = b; // b and c are alias
```



References:3

Objects:2

--

Garbage Collection

When an object no longer has any valid references to it, it can no longer be accessed by the program

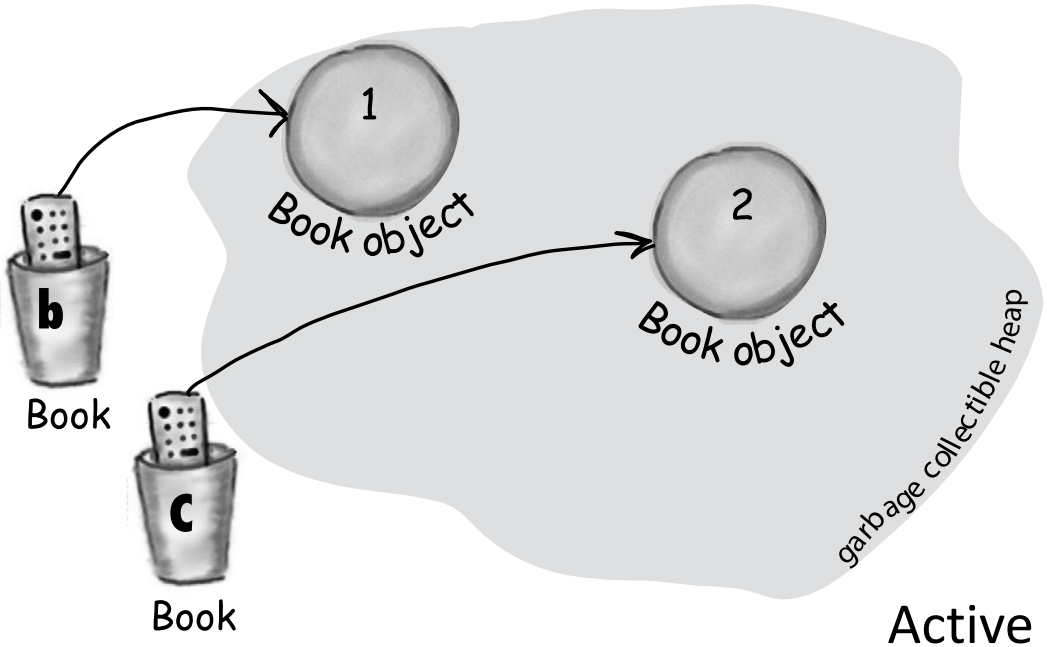
The object is useless, and therefore is called *garbage*

Java performs *automatic garbage collection* periodically, returning an object's memory to the system for future use

In other languages, the programmer is responsible for performing garbage collection

Example

```
Book b = new Book();
Book c = new Book();
```

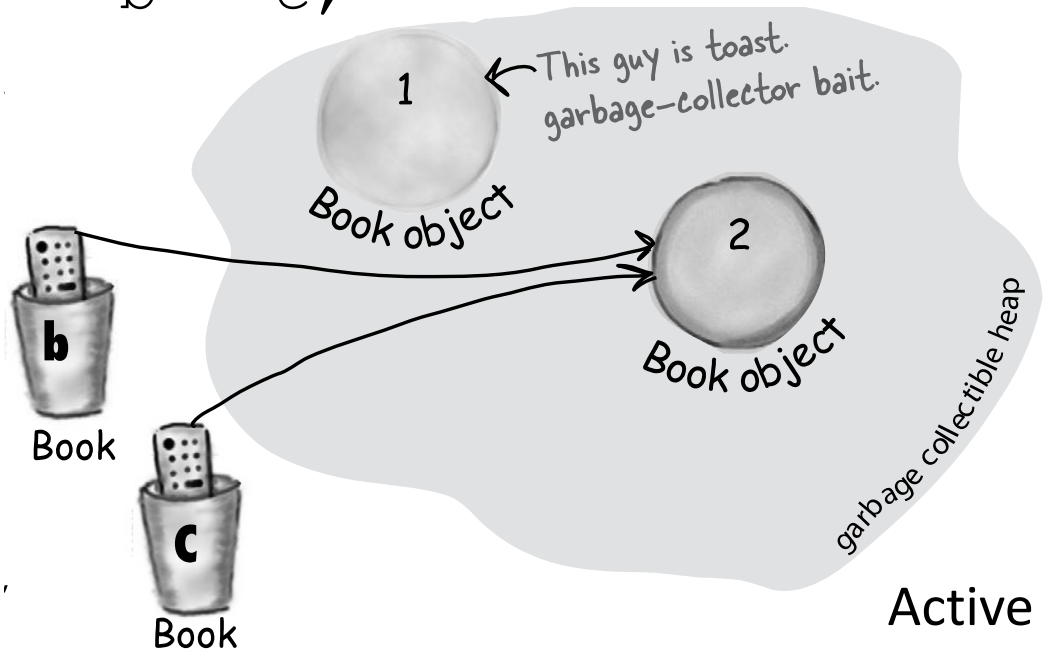


Active references: 2

Reachable objects:2

Example

```
Book b = new Book();
Book c = new Book();
b = c;
```



Active references: 2

Reachable objects:1

Abandoned objects: 1

The Math Class

The `Math` class is part of the `java.lang` package

The `Math` class contains methods that perform various mathematical functions

These include:

- absolute value
- square root
- exponentiation
- trigonometric functions

The Math Class

The methods of the Math class are ***static methods*** (also called ***class methods***)

Static methods are invoked through the class name – no object of the Math class is needed

value = Math.cos(90) + Math.sqrt(delta);

Common methods in the Math class

```
Math.abs(num)          x = Math.abs(num);

Math.sqrt(num)         x = Math.sqrt(discriminant);

Math.pow(num, power)   x = Math.pow(b, 2);
```

```
//********************************************************************
//  Quadratic.java       Author: Lewis/Loftus
//
//  Demonstrates the use of the Math class to perform a calculation
//  based on user input.
//********************************************************************

import java.util.Scanner;

public class Quadratic
{
   //-----------------------------------------------------------------
   //  Determines the roots of a quadratic equation.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      int a, b, c;  // ax^2 + bx + c
      double discriminant, root1, root2;

      Scanner scan = new Scanner (System.in);

      System.out.print ("Enter the coefficient of x squared: ");
      a = scan.nextInt();
```

continued

continued

```
      System.out.print ("Enter the coefficient of x: ");
      b = scan.nextInt();

      System.out.print ("Enter the constant: ");
      c = scan.nextInt();

      // Use the quadratic formula to compute the roots.
      // Assumes a positive discriminant.

      discriminant = Math.pow(b, 2) - (4 * a * c);
      root1 = ((-1 * b) + Math.sqrt(discriminant)) / (2 * a);
      root2 = ((-1 * b) - Math.sqrt(discriminant)) / (2 * a);

      System.out.println ("Root #1: " + root1);
      System.out.println ("Root #2: " + root2);
   }
}
```

continued

```
        System
        b = sc

        System
        c = sc

        // Use the quadratic formula to compute the roots.
        // Assumes a positive discriminant.

        discriminant = Math.pow(b, 2) - (4 * a * c);
        root1 = ((-1 * b) + Math.sqrt(discriminant)) / (2 * a);
        root2 = ((-1 * b) - Math.sqrt(discriminant)) / (2 * a);

        System.out.println ("Root #1: " + root1);
        System.out.println ("Root #2: " + root2);
    }
}
```

Sample Run

```
Enter the coefficient of x squared: 3
Enter the coefficient of x: 8
Enter the constant: 4
Root #1: -0.6666666666666666
Root #2: -2.0
```

The Random Class

The `Random` class is part of the `java.util` package

It provides methods that generate pseudorandom numbers

A `Random` object performs complicated calculations based on a seed value to produce a stream of seemingly random values

- nextFloat() generates a random number [0, 1)
- nextInt() generates a random integer (positive or negative)
- nextInt(n) generates a random integer [0, n-1]

```
//********************************************************************
//  RandomNumbers.java       Author: Lewis/Loftus
//
//  Demonstrates the creation of pseudo-random numbers using the
//  Random class.
//********************************************************************

import java.util.Random;

public class RandomNumbers
{
   //-----------------------------------------------------------------
   //  Generates random numbers in various ranges.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      Random generator = new Random();
      int num1;
      float num2;

      num1 = generator.nextInt();
      System.out.println ("A random integer: " + num1);

      num1 = generator.nextInt(10);
      System.out.println ("From ? to ?: " + num1);
```

continued

continued

```
        num1 = generator.nextInt(10) + 1;
        System.out.println ("From ? to ?: " + num1);

        num1 = generator.nextInt(15) + 20;
        System.out.println ("From ? to ?: " + num1);

        num1 = generator.nextInt(20) - 10;
        System.out.println ("From ? to ?: " + num1);

        num2 = generator.nextFloat();
        System.out.println ("A random float (between 0-1): " + num2);

        num2 = generator.nextFloat() * 6;
        num1 = (int)num2 + 1;
        System.out.println ("From ? to ?: " + num1);
    }
}
```

continued

```
        num1
        Syst

        num1
        Syst

        num1
        Syst

        num2 = generator.nextFloat();
        System.out.println ("A random float (between 0-1): " + num2);

        num2 = generator.nextFloat() * 6;  // 0.0 to 5.999999
        num1 = (int)num2 + 1;
        System.out.println ("From 1 to 6: " + num1);
    }
}
```

Sample Run

```
A random integer: 672981683
From 0 to 9: 0
From 1 to 10: 3
From 20 to 34: 30
From -10 to 9: -4
A random float (between 0-1): 0.18538326
From 1 to 6: 3
```

Another Way to Generate a Random Number

```
double newRand = Math.random();
```

generates a random double number [0,1)

as opposed to

```
import java.util.Random;
 Random generator = new Random();
 float newRand = generator.nextFloat();
```

Enumerated Types

Java allows you to define an ***enumerated type***, which can then be used to declare variables

An enumerated type declaration lists all possible values for a variable of that type

The values are identifiers of your own choosing

The following declaration creates an enumerated type called `Season`

```
enum Season {winter, spring, summer, fall};
```

Any number of values can be listed

Enumerated Types

Once a type is defined, a variable of that type can be declared:

```
Season time;
```

And it can be assigned a value:

```
time = Season.fall;
```

The values are referenced through the name of the type

Enumerated types are ***type-safe*** – you cannot assign any value other than those listed

Enumerated Types

The declaration of an enumerated type is a special type of class, and each variable of that type is an object

The `ordinal` method returns the ordinal value of the object

The first value in an enumerated type has an ordinal value of 0, the second 1, and so on

The `name` method returns the name of the identifier corresponding to the object's value

```
//********************************************************************
//  IceCream.java       Author: Lewis/Loftus
//
//  Demonstrates the use of enumerated types.
//********************************************************************

public class IceCream
{
   enum Flavor {vanilla, chocolate, strawberry, fudgeRipple, coffee,
                rockyRoad, mintChocolateChip, cookieDough}

   //-----------------------------------------------------------------
   //  Creates and uses variables of the Flavor type.
   //-----------------------------------------------------------------
   public static void main (String[] args)
   {
      Flavor cone1, cone2, cone3;

      cone1 = Flavor.rockyRoad;
      cone2 = Flavor.chocolate;

      System.out.println ("cone1 value: " + cone1);
      System.out.println ("cone1 ordinal: " + cone1.ordinal());
      System.out.println ("cone1 name: " + cone1.name());
```

continued

continued

```
        System.out.println ();
        System.out.println ("cone2 value: " + cone2);
        System.out.println ("cone2 ordinal: " + cone2.ordinal());
        System.out.println ("cone2 name: " + cone2.name());

        cone3 = cone1;

        System.out.println ();
        System.out.println ("cone3 value: " + cone3);
        System.out.println ("cone3 ordinal: " + cone3.ordinal());
        System.out.println ("cone3 name: " + cone3.name());
    }
}
```

continued

```
        System.out.prin
        System.out.prin                                    );
        System.out.prin                                 e2.ordinal());
        System.out.prin                                 name());

        cone3 = cone1;

        System.out.prin
        System.out.prin                                    );
        System.out.prin                                 e3.ordinal());
        System.out.println ("cone3 name: " + cone3.name());
    }
}
```

Output

```
cone1 value: rockyRoad
cone1 ordinal: 5
cone1 name: rockyRoad
cone2 value: chocolate
cone2 ordinal: 1
cone2 name: chocolate
cone3 value: rockyRoad
cone3 ordinal: 5
cone3 name: rockyRoad
```

Group Exercises

Ex: 3.7

Ex: 3.8

Ex: 3.9

Ex: 3.13

Assignment for Class 7

Review RandomNumbers, Quadratic, Purchase, CircleStats, IceCream

Read Chapter 3.9, 3.10, 3.11