6.4 Overloading

- **The same method name** has more than one definition *within* [redacted] *class*
- Each definition must have [redacted]
  - » different argument types, a different number of arguments, or a different ordering of argument types
  - » [redacted] is **not** part of the [redacted] and **cannot** be used to distinguish between two methods with the same name and parameter types

Listing 6.15 Overloading

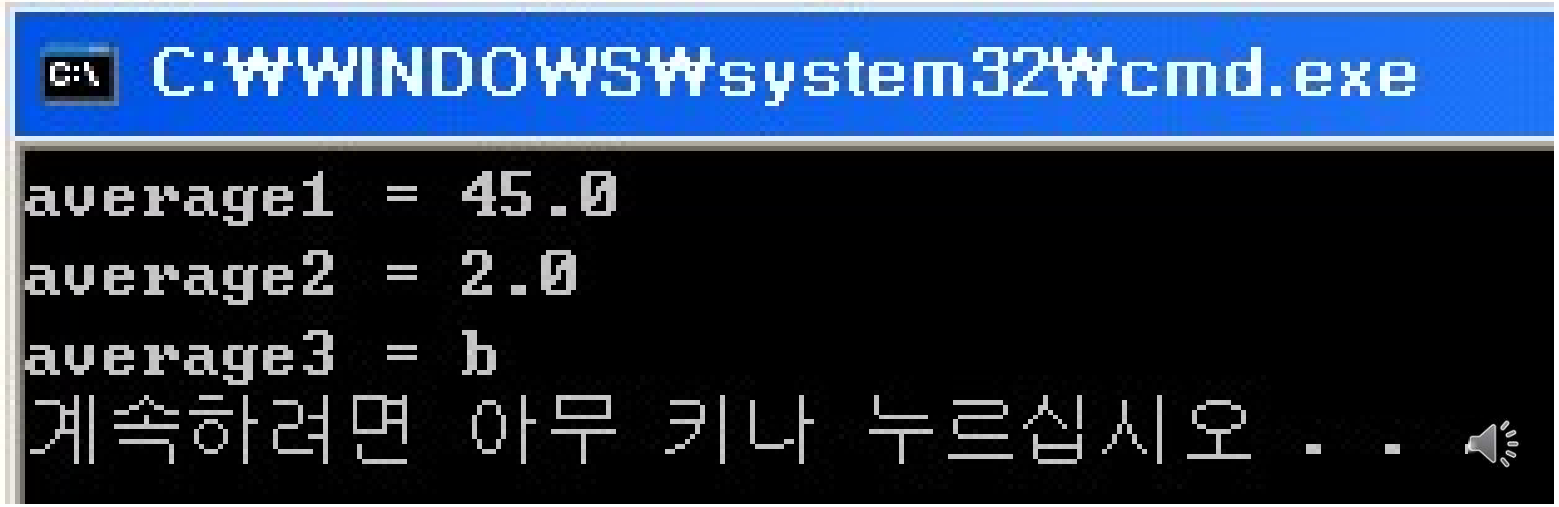
```
/**
 This class illustrates overloading.
*/
public class Overload
{
   public static void main(String[] args)
   {
      double average1 = Overload.getAverage(40.0, 50.0);
      double average2 = Overload.getAverage(1.0, 2.0, 3.0);
      char   average3 = Overload.getAverage('a', 'c');

      System.out.println("average1 = " + average1);
      System.out.println("average2 = " + average2);
      System.out.println("average3 = " + average3);
   }
```

```
public static double getAverage(double first, double second)
  {
     return (first + second) / 2.0;
  }

  public static double getAverage(double first, double second,
                                  double third)
  {
     return (first + second + third) / 3.0;
  }

  public static char getAverage(char first, char second)
  {
     return (char)(((int)first + (int)second) / 2);
  }
}
```



Signature

- [redacted]
- `equals(Species)` has a different signature than `equals(String)`
  - » same method name, <u>different argument types</u>
- *`myMethod(1)`* has a different signature than *`myMethod(1, 2)`*
  - » same method name, different number of arguments
- *`myMethod(10, 1.2)`* has a different signature than *`myMethod(1.2, 10)`*
  - » same method name and number of arguments, but different order of argument types

DISPLAY 5.16 Class Diagram for Pet Class

```
Pet
-------------------------------------------------------------
- name: String
- age: int
- weight: double
-------------------------------------------------------------
+ writeOutput( ): void
+ set(String newName): void
+ set(int newAge): void
+ set(double newWeight): void
+ set(String newName, int newAge, double newWeight): void
+ getName( ): String
+ getAge( ): int
+ getWeight( ): double
```

Display Pet Class - Pet.java

```
// Display  Pet Class

/**
 Class for basic pet records: name, age, and weight.
*/
public class Pet
{
    private String name;
    private int age; //in years
    private double weight; //in pounds
```

```
/**
 This main is just a demonstration program.
*/
public static void main(String[] args)
{
    Pet myDog = new Pet( );
    myDog.set("Fido", 2, 5.5);
    myDog.writeOutput( );
    System.out.println("Changing name.");
    myDog.set("Rex");
    myDog.writeOutput( );
    System.out.println("Changing weight.");
    myDog.set(6.5);
    myDog.writeOutput( );
    System.out.println("Changing age.");
    myDog.set(3);
    myDog.writeOutput( );
}
```

```
public void writeOutput( )
 {
    System.out.println("Name: " + name);
    System.out.println("Age: " + age + " years");
    System.out.println("Weight: " + weight + " pounds");
 }

 public void set(String newName)  //overload
 {
    name = newName;
    //age and weight are unchanged.
 }

 public void set(int newAge) //overload
 {
  ..............................
 }

 public void set(double newWeight) //overload
 {
    ..............................
 }
```

```
 public void set(String newName, int newAge, double
newWeight)//overload
   {
      name = newName;
      if ((newAge <= 0) || (newWeight <= 0))
      {
         System.out.println("Error: illegal age or weight.");
         System.exit(0);
      }
      else
      {
         age = newAge;
         weight = newWeight;
      }
   }
      public String getName( )
      {       return name;    }
      public int getAge( )
      {           return age;       }
       public double getWeight( )
      {           return weight;       }
}
```

C:₩WINDOWS₩system32₩cmd.exe

```
Name: Fido
Age: 2 years
Weight: 5.5 pounds
Changing name.
Name: Rex
Age: 2 years
Weight: 5.5 pounds
Changing weight.
Name: Rex
Age: 2 years
Weight: 6.5 pounds
Changing age.
Name: Rex
Age: 3 years
Weight: 6.5 pounds
계속하려면 아무 키나 누르십시오 . . .
```

Overloading and Argument Type

- Accidentally using the wrong datatype as an argument can invoke a different method

- For example, see the `Pet` class in the text
  - » `set(int)` sets the pet's age
  - » `set(double)` sets the pet's weight
  - » You want to set the pet's weight to 6 pounds:
    - – `set(6.0)` works as you want because the argument is type `double`
    - – `set(6)` will set the *age* to 6, not the weight, since the argument is type **`int`**

Gotcha: Overloading and Automatic Type Conversion

- If Java does not find a signature match, it attempts some automatic type conversions, e.g. `int` to `double`
  - » An unwanted version of the method may execute
- In the text Pet example of overloading:
  (PetTest.java) What you want: name "Cha Cha", weight 2, and age 3
  But you make two mistakes:
  1. you reverse the age and weight numbers, and
  2. you fail to make the weight a type double.
  - » `set("Cha Cha", 2, 3)` does not do what you want
    - – it sets the pet's age = 2 and the weight = 3.0
  - » Why?

    컴파일러마다 차이가 O
    - – `set` has no definition with the argument types `String`, `int`, `int`
    - – However, it does have a definition with `String`, `int`, `double`, so it promotes the last number, 3, to 3.0 and executes the method with that signature
- In other situations, automatic type conversions can make method invocations ambiguous.

Overload
Based on the Returned Type

```
// Math Test

public class OverloadReturnTypeTest
{
        public double getWeight()  { }
        public char getWeight() { }

}
```

Gotcha: You Cannot Overload Based on the Returned Type

- Compiler will not allow two methods with same name, same types and number of parameters, but different return types in the same class:

```
public double getWeight()
```

```
public char getWeight()
```

Can't have both in the same class

- In a situation like this you would have to change the name of one method or change the number or types of parameters.

- 패러미터 타입이 같은데 리턴타입이 다르다 (X)
- 패러미터 타입이 다른데 리턴 타입이 같다 (ㅇ)
- 패러미터 타입도 다르고 리턴 타입도 다르다(ㅇ)

```
public class OverloadReturnTypeTest3  //OK
{
        public double getWeight()  { return 0.0; }
        public double getWeight(int a) { return 0.0; }
}
```

```
public class OverloadReturnTypeTest  // OK!!
{
        public double getWeight()  { return 0.0; }
        public char getWeight(int a) { return 'a'; }
}
```

- 패러미터 타입이 같고, 리턴 타입이 같다, 패러미터의 이름이 다르다. (X)

```
public class OverloadReturnTypeTest4  //No
{
        public double getWeight(int b)  { return 0.0; }
        public double getWeight(int a) { return 0.0; }
}
```

// Listing 6.16

```
import java.util.Scanner;
/**
Class representing nonnegative amounts of money,
such as $100, $41.99, $0.05.
*/
public class Money
{
    private long dollars;
    private long cents;
    public void set (long newDollars)
    {
        if (newDollars < 0)
        {
            System.out.println ("Error: Negative amounts of money are not allowed.");
            System.exit (0);
        }
        else
        {
            dollars = newDollars;
            cents = 0;
        }
    }
```

```
public void set (double newAmount)
{
    if (newAmount < 0)
    {
        System.out.println (
                "Error: Negative amounts of money are not allowed.");
        System.exit (0);
    }
    else
    {
        long allCents = Math.round (newAmount * 100);
        dollars = allCents / 100;
        cents = allCents % 100;
    }
}


public void set (Money moneyObject)
{
    this.dollars = moneyObject.dollars;
    this.cents = moneyObject.cents;
}
```

```
/**
Precondition: The argument is an ordinary representation
of an amount of money, with or without a dollar sign.
Fractions of a cent are not allowed.
*/
public void set (String amountString)
{
    String dollarsString;
    String centsString;
    //Delete '$' if any:
    if (amountString.charAt (0) == '$')
        amountString = amountString.substring (1);
    amountString = amountString.trim ();
    //Locate decimal point:
    int pointLocation = amountString.indexOf (".");
    if (pointLocation < 0) //If no decimal point
    {
        cents = 0;
        dollars = Long.parseLong (amountString);
    }
```

```
    else //String has a decimal point.
    {
        dollarsString =
            amountString.substring (0, pointLocation);
        centsString =
            amountString.substring (pointLocation + 1);
        //one digit in cents means tenths of a dollar
        if (centsString.length () <= 1)
            centsString = centsString + "0";
        // convert to numeric
        dollars = Long.parseLong (dollarsString);
        cents = Long.parseLong (centsString);
        if ((dollars < 0) || (cents < 0) || (cents > 99))
        {
            System.out.println (
                    "Error: Illegal representation of money.");
            System.exit (0);
        }
    }
}
```

```
 public void readInput ()
{
    System.out.println ("Enter amount on a line by itself:");
    Scanner keyboard = new Scanner (System.in);
    String amount = keyboard.nextLine ();
    set (amount.trim ());
}


/**
Does not go to the next line after displaying money.
*/
public void writeOutput ()
{
    System.out.print ("$" + dollars);
    if (cents < 10)
        System.out.print (".0" + cents);
    else
        System.out.print ("." + cents);
}
```

```
/**    Returns n times the calling object.    */
   public Money times (int n)
   {
      Money product = new Money ();
      product.cents = n * cents;
      long carryDollars = product.cents / 100;
      product.cents = product.cents % 100;
      product.dollars = n * dollars + carryDollars;
      return product;
   }
   /**    Returns the sum of the calling object and the argument.    */
   public Money add (Money otherAmount)
   {
      Money sum = new Money ();
      sum.cents = this.cents + otherAmount.cents;
      long carryDollars = sum.cents / 100;
      sum.cents = sum.cents % 100;
      sum.dollars = this.dollars
         + otherAmount.dollars + carryDollars;
      return sum;
   }
}
```

// Listing 6.17

```
public class MoneyDemo
{
    public static void main (String [] args)
    {
        Money start = new Money ();
        Money goal = new Money ();
        System.out.println ("Enter your current savings:");
        start.readInput ();
        goal = start.times (2);
        System.out.print (
                "If you double that, you will have ");
        goal.writeOutput ();
        System.out.println (", or better yet:");
        goal = start.add (goal);
        System.out.println (
                "If you triple that original amount, you will have:");
        goal.writeOutput ();
        System.out.println ();
        System.out.println ("Remember: A penny saved");
        System.out.println ("is a penny earned.");
    }
}
```

C:₩WINDOWS₩system32₩cmd.exe

```
Enter your current savings:
Enter amount on a line by itself:
2000
If you double that, you will have $4000.00, or better yet:
If you triple that original amount, you will have:
$6000.00
Remember: A penny saved
is a penny earned.
계속하려면 아무 키나 누르십시오 . . .
```

끝