CS5000: Foundations of Programming

Mingon Kang, PhD
Computer Science, Kennesaw State University
Overloading

- Two or more methods in the same class can have same method name with different parameters
- Called “Overloaded functions”
- Ambiguous method invocations will produce an error in JAVA
public void PrintValues(int nValue1)
{
    ...
}
public void PrintValues(int nValue1, int nValue2)
{
    ...
}
public void Print()
{
    PrintValues(1);
    PrintValues(3, 4);
}
public void Add(int nValue)
{
    ...
}
public void Add(float nValue)
{
    ...
}
public static void main(String[] args)
{
    Add(1);
    Add(3.4);
}
Sometimes there may be two or more possible matches for an invocation of a method, but the compiler cannot determine the most specific match. This is referred to as *ambiguous invocation*. Ambiguous invocation is a compile error.

Ref: Liang, *Introduction to Java Programming*
public class AmbiguousOverloading {
    public static void main(String[] args) {
        System.out.println(max(1, 2));
    }

    public static double max(int num1, double num2) {
        if (num1 > num2)
            return num1;
        else
            return num2;
    }

    public static double max(double num1, int num2) {
        if (num1 > num2)
            return num1;
        else
            return num2;
    }
}
Examples

- Write a method that display (or return) the grade for a given score

  ```java
  public static void printGrade(double score)
  public static char printGrade(double score)
  
  Also Check if the score is valid or nor
Write a method of “increment” that the parameter is incremented by 1.

```java
public static void main(String[] args) {
    int x = 1;
    increment(x); // Does it work?
    System.out.println(x);
}
```
Examples

- Write a method to display an identity matrix of \( n \times n \) square matrix

\[
I_1 = [1], \quad I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \ldots, \quad I_n = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}
\]
Examples

- Generate a random number between $a$ and $b$
  - Shall use the method again later when needed
  - Implement the method as an independent module from others.

```java
public static int generateRandom(int nMin, int nMax)
```
Examples

- (Estimate $\pi$) $\pi$ can be computed using the following series:

$$m(i) = 4 \left( 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} + \cdots + \frac{(-1)^{i+1}}{2i - 1} \right)$$

Write a method that returns $m(i)$ for a given $i$ and write a test program that displays the following table:

<table>
<thead>
<tr>
<th>i</th>
<th>m(i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>101</td>
<td>3.1515</td>
</tr>
<tr>
<td>301</td>
<td>3.1449</td>
</tr>
<tr>
<td>501</td>
<td>3.1436</td>
</tr>
<tr>
<td>801</td>
<td>3.1428</td>
</tr>
<tr>
<td>901</td>
<td>3.1427</td>
</tr>
</tbody>
</table>
Benefits of methods

- Improving readability
- Reusing methods
- Easy to develop, debug, and test
- Better facilitating teamwork