دانشکده مهندسی و علوم کامپیوتر

برنامه نویسی پیشرفته

وحیدی اصل

آشنایی با متدها

انگیزه-چرا به متدها نیاز داریم!

برنامه ای بنویسید که مجموع اعداد صحیح به ترتیب از 1 تا 10، از 20 تا 30 و از 35 تا 45 را محاسبه کند.

انگیزه-چرا به متدها نیاز داریم!

```
int sum = 0;
for (int i = 1; i <= 10; i++)
  sum += i;
System.out.println("Sum from 1 to 10 is " + sum);

sum = 0;
for (int i = 20; i <= 30; i++)
  sum += i;
System.out.println("Sum from 20 to 30 is " + sum);

sum = 0;
for (int i = 35; i <= 45; i++)
  sum += i;
System.out.println("Sum from 35 to 45 is " + sum);
```

انگیزه-چرا به متدها نیاز داریم!

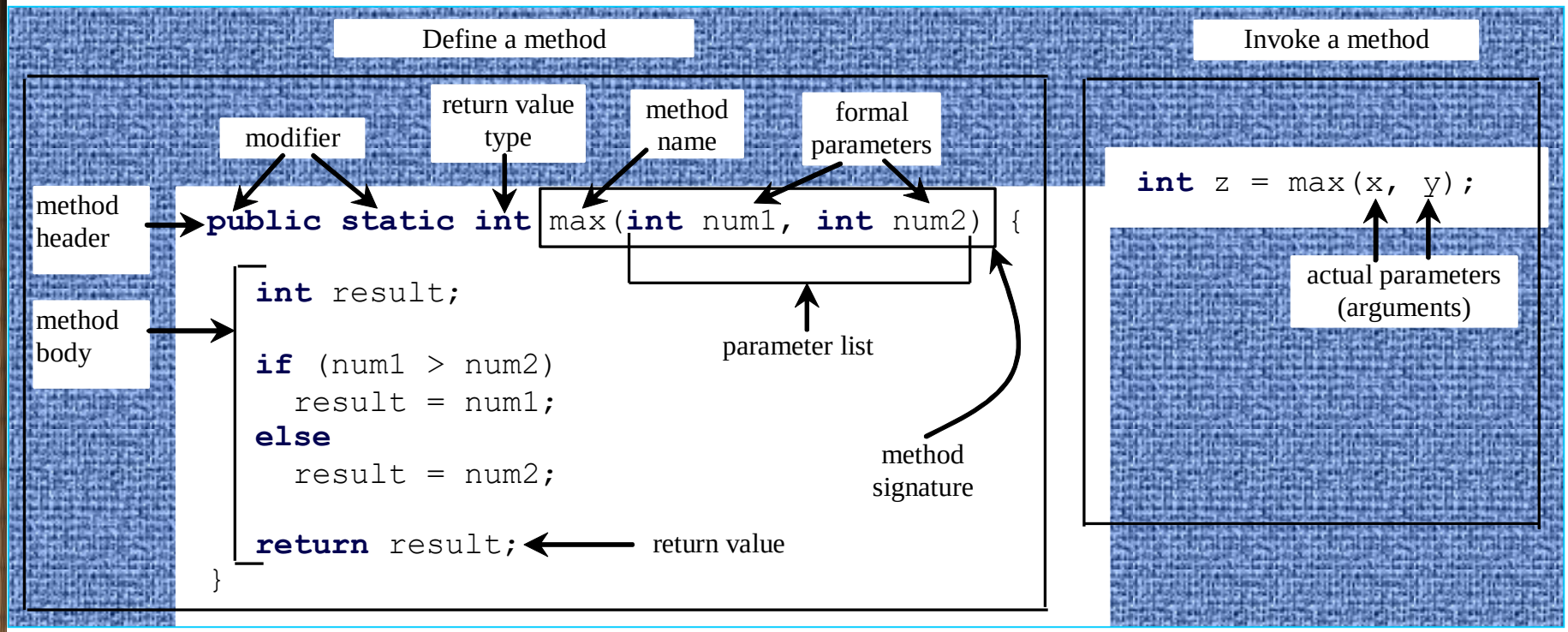
```
int sum = 0;
for (int i = 1; i <= 10; i++)
  sum += i;
System.out.println("Sum from 1 to 10 is " + sum);

sum = 0;
for (int i = 20; i <= 30; i++)
  sum += i;
System.out.println("Sum from 20 to 30 is " + sum);

sum = 0;
for (int i = 35; i <= 45; i++)
  sum += i;
System.out.println("Sum from 35 to 45 is " + sum);
```

راه حل-استفاده از متدها!

```
public static int sum(int i1, int i2) {
  int sum = 0;
  for (int i = i1; i <= i2; i++)
    sum += i;
  return sum;
}

public static void main(String[] args) {
  System.out.println("Sum from 1 to 10 is " + sum(1, 10));
  System.out.println("Sum from 20 to 30 is " + sum(20, 30));
  System.out.println("Sum from 35 to 45 is " + sum(35, 45));
}
```

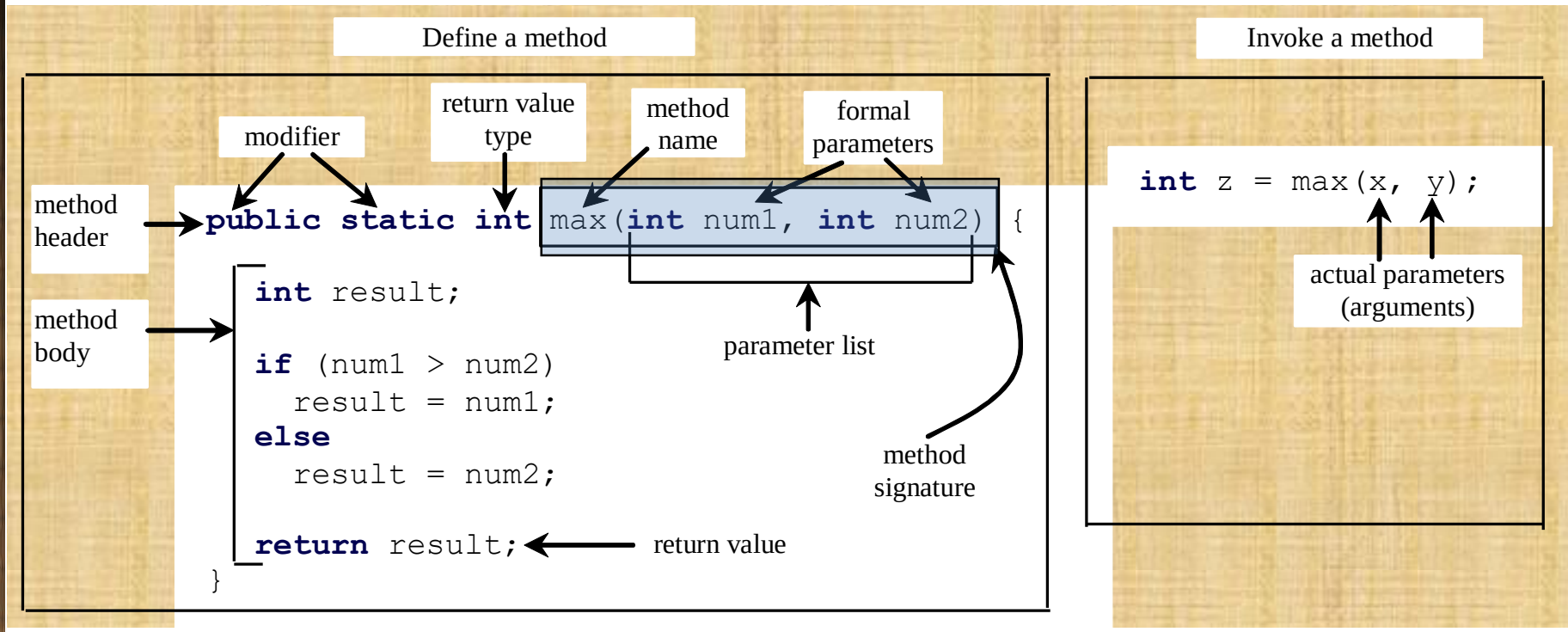
تعریف متدها

- یک متد مجموعه ای دستورات است که به همراه یکدیگر یک یا چند عمل را انجام می دهند.



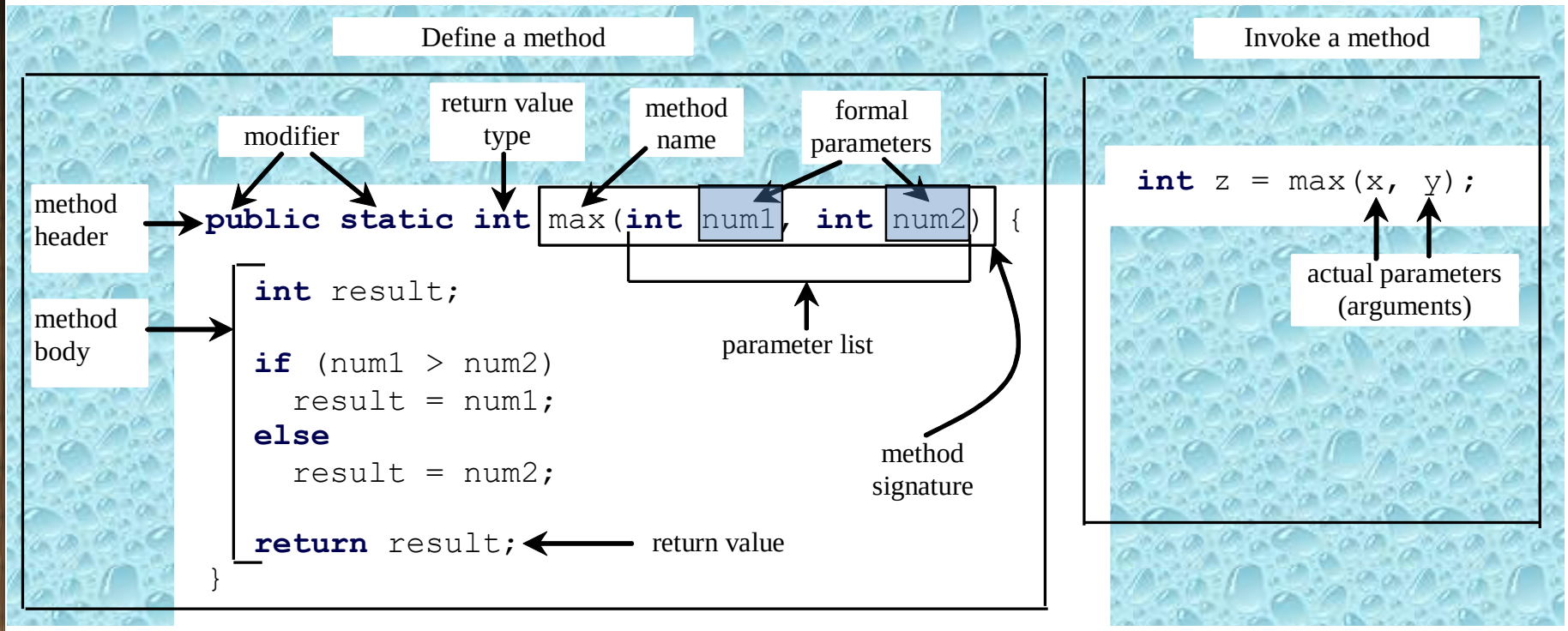
امضای متد(method signature)

- امضای متد ترکیبی از نام متد و لیست پارامترهای آن می باشد.



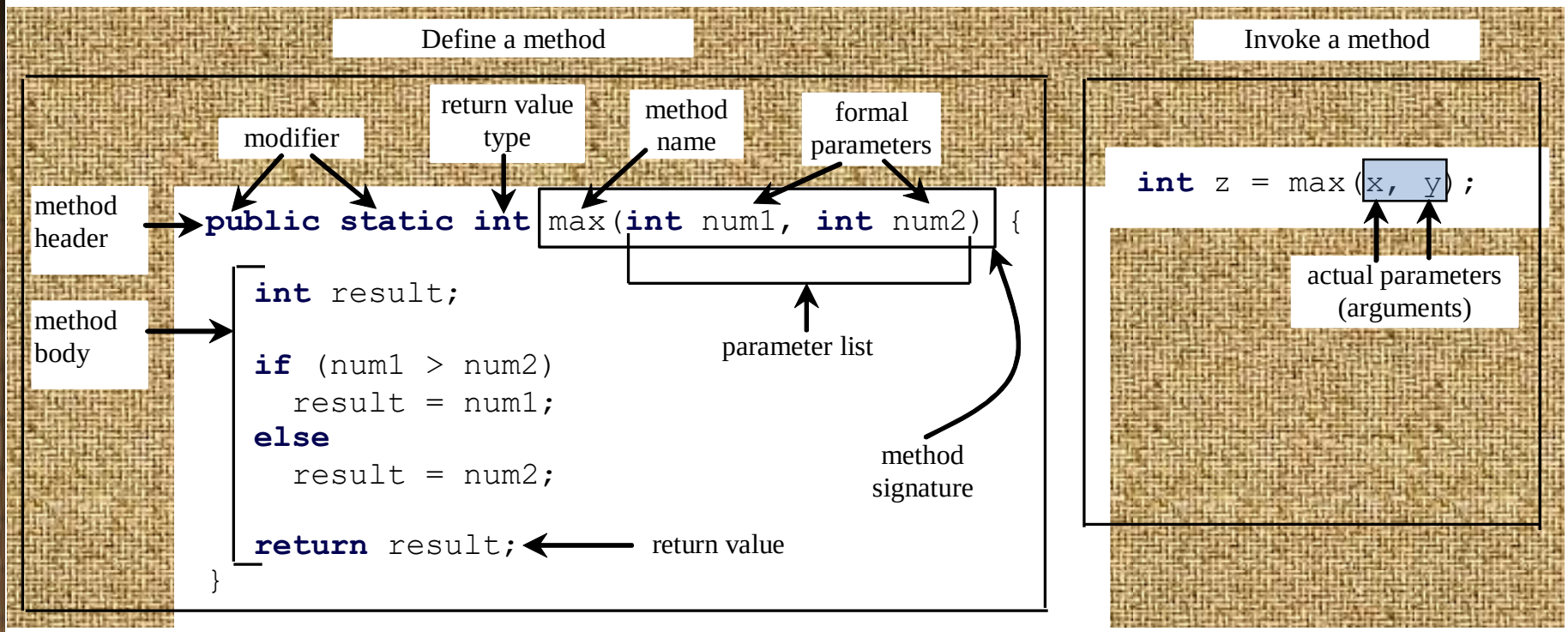
پارامترهای فرمال

•متغیرهایی که در لیست پارامترهای متد تعریف شده اند، پارامترهای فرمال (صوری) نامیده می شوند.



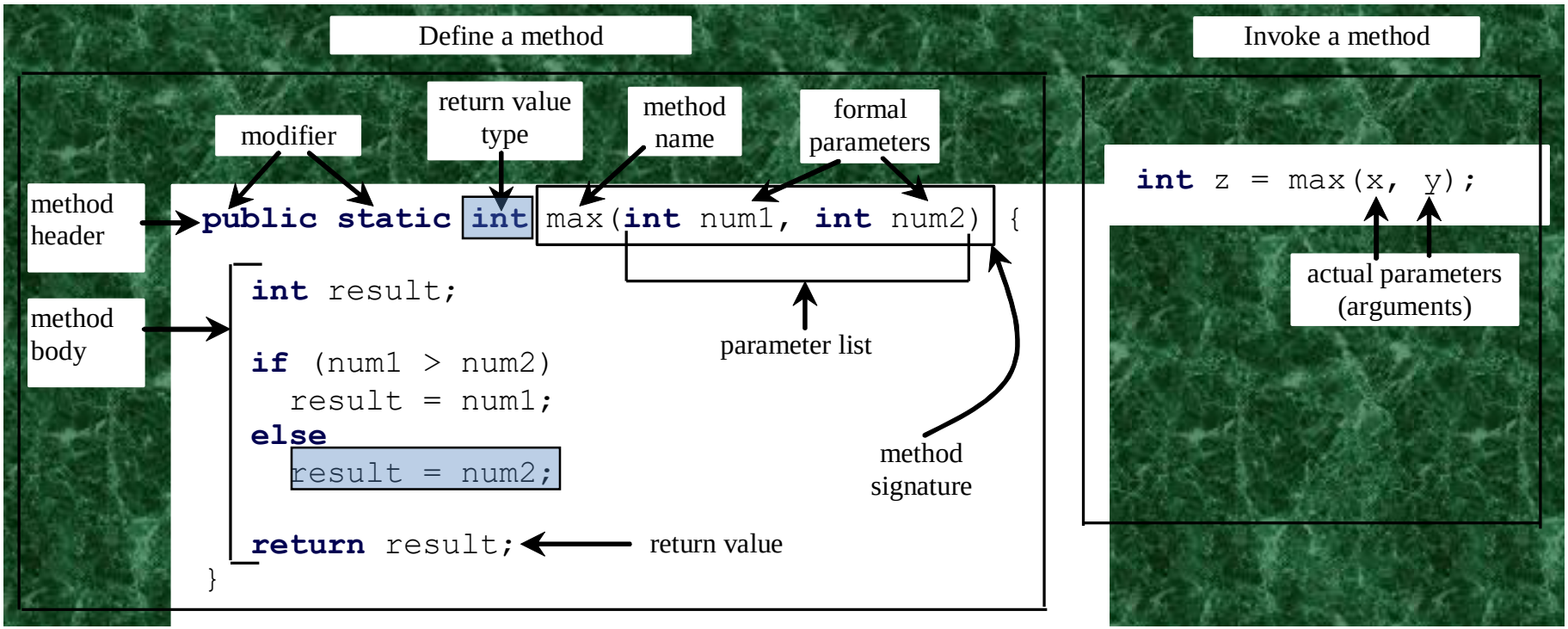
پارامترهای واقعی

- هرگاه متدی فراخوانی می شود، شما مقداری را به پارامتر آن ارسال می کنید. به این مقدار اصطلاحاً پارامتر واقعی (actual parameter) یا آرگومان گفته می شود.



نوع مقدار برگشتی

- یک متد ممکن است مقداری را برگرداند.
- نوع مقدار برگشتی نوع داده ای است که متد به محلی که از آن فراخوانده شده بر می گرداند.
- اگر متدی مقداری برنگرداند، نوع مقدار برگشتی void خواهد بود.
- برای مثال نوع برگشتی تابع main ، void می باشد

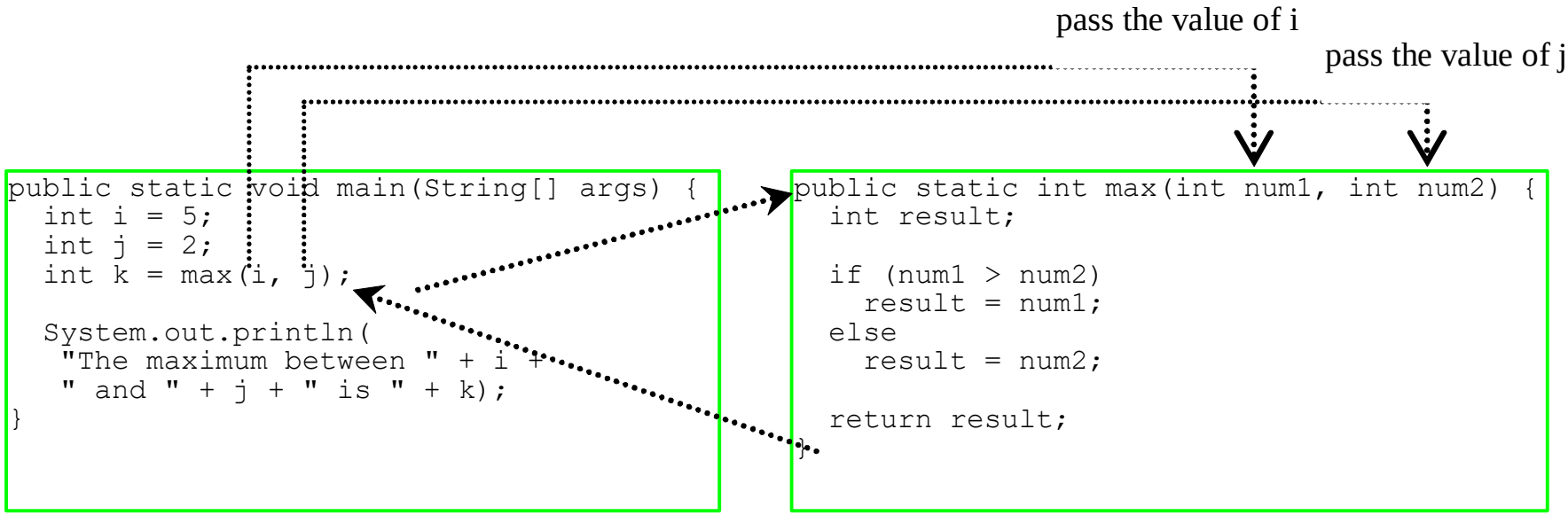


فراخوانی متدها

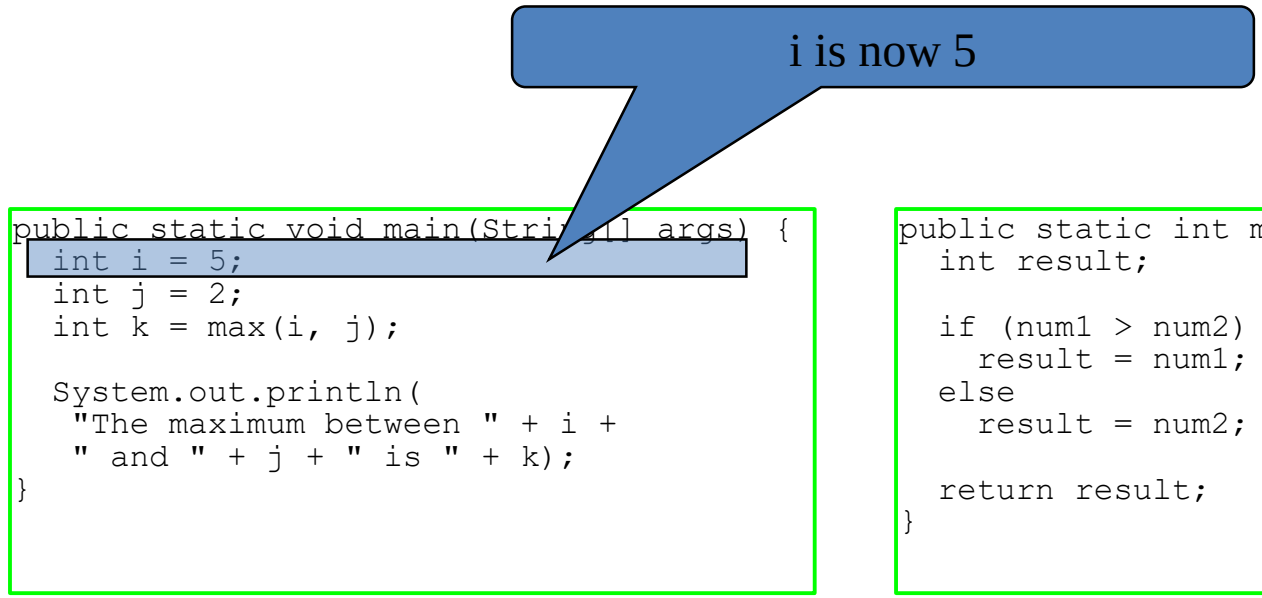
- برنامه نشان داده شده در اسلاید بعدی یک متد max را فراخوانی می کند تا بزرگترین مقدار از میان مقادیر int را برگرداند.

```
public class TestMax {
  /** Main method */
  public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);
    System.out.println("The maximum between " + i +  " and " + j + " is " + k);
  }

  /** Return the max between two numbers */
  public static int max(int num1, int num2) {
    int result;

    if (num1 > num2)
      result = num1;
    else
      result = num2;

    return result;
  }
}
```

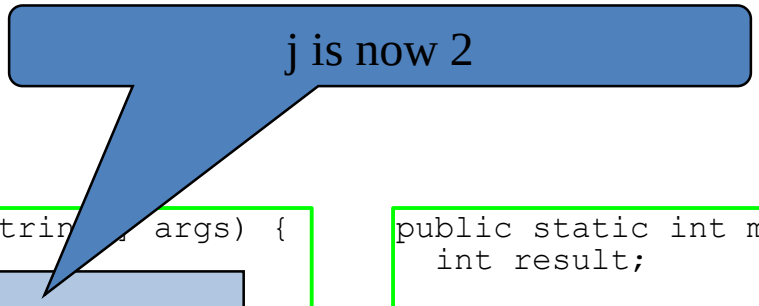
فراخوانی متدها-ادامه



ردگیری فراخوانی متد



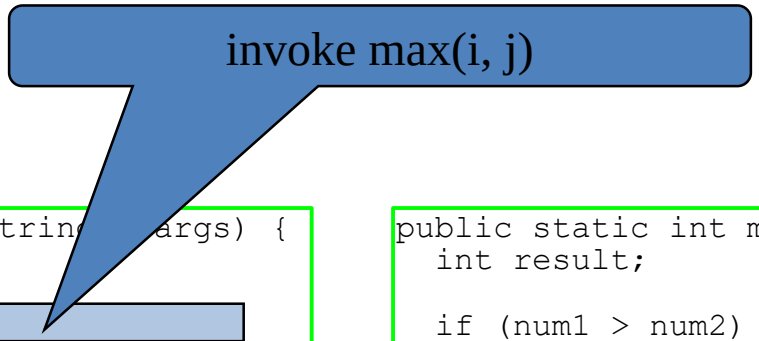
ردگیری فراخوانی متد




```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

ردگیری فراخوانی متد




```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

ردگیری فراخوانی متد

invoke max(i, j)
Pass the value of i to num1
Pass the value of j to num2

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

ردگیری فراخوانی متد

declare variable result

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

ردگیری فراخوانی متد

(num1 > num2) is true since num1 is 5 and num2 is 2

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

ردگیری فراخوانی متد

result is now 5

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

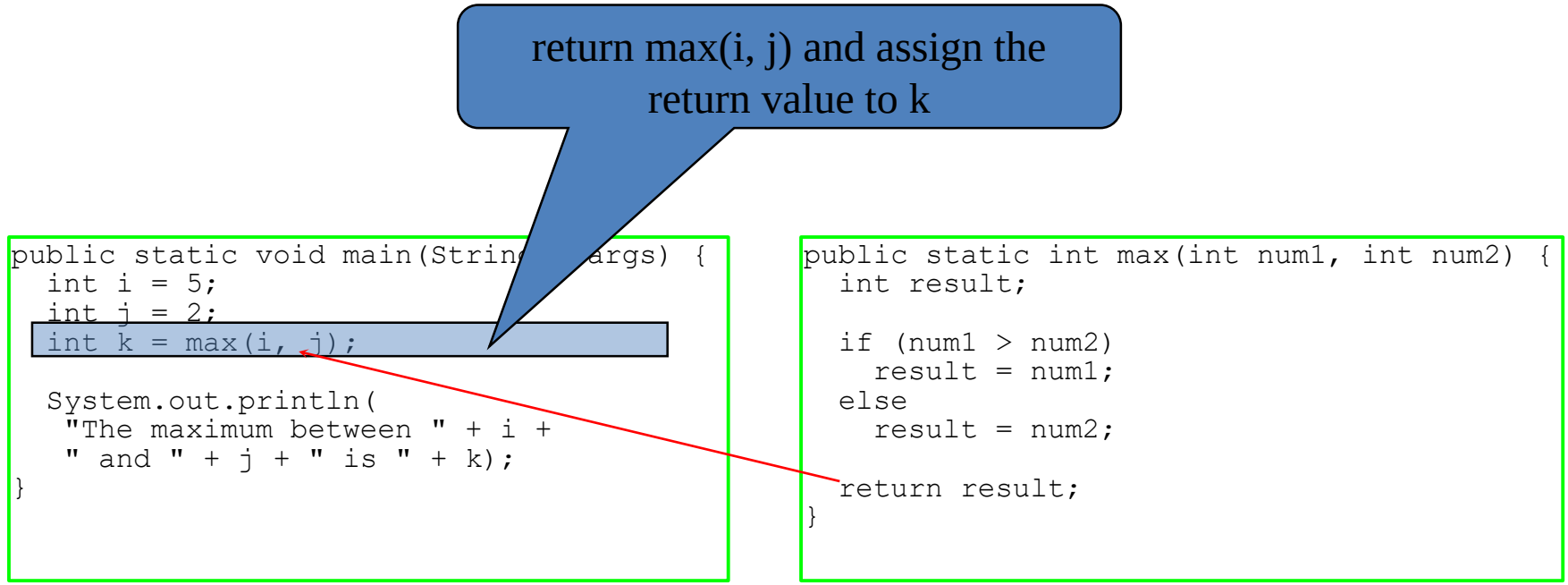
ردگیری فراخوانی متد

return result, which is 5

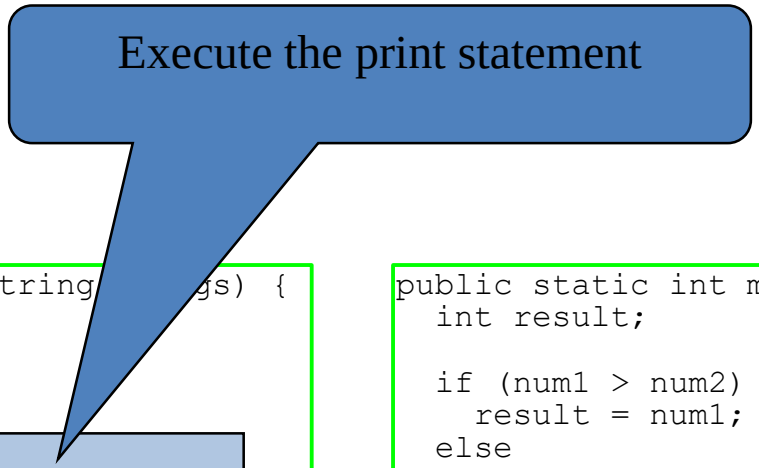
```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
pu    tatic int max(int num1, int num2) {
      esult;

      num1 > num2)
      esult = num1;
    se
      result = num2;

  return result;
}
```

ردگیری فراخوانی متد



ردگیری فراخوانی متد




```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

دقت کنید!

- در متدهایی که مقدار برگشتی دارند، به یک دستور return برای برگرداندن مقدار برگشتی نیاز می باشد.
- متد نشان داده شده در(a) از لحاظ منطقی صحیح است، اما از نظر کامپایلردارای خطای زمان کامپایل می باشد.
  - چون کامپایلر تصور می کند، این امکان وجود دارد که این متد هیچ مقداری را برنگرداند.

```
public static int sign(int n) {
  if (n > 0)
    return 1;
  else if (n == 0)
    return 0;
  else if (n < 0)
    return -1;
}
```

(a)

Should be

```
public static int sign(int n) {
  if (n > 0)
    return 1;
  else if (n == 0)
    return 0;
  else
    return -1;
}
```

(b)

- برای رفع این مشکل، کافسیت *if (n < 0)* در (a)را پاک کنید. در نتیجه کامپایلر خواهد فهمید که دستور return بدون توجه به مقدار شرطی، نهایتاً مقداری را برخواهد گرداند.

استفاده مجدد متدها از کلاسهای دیگر

- نکته:
- یکی از مزایای متدها استفاده مجدد از آنها در برنامه های مختلف است.
- متد max می تواند از کلاسهای مختلف فراخوانی شود:

ClassName.methodName (e.g., TestMax.max).

پشته فراخوانی ها

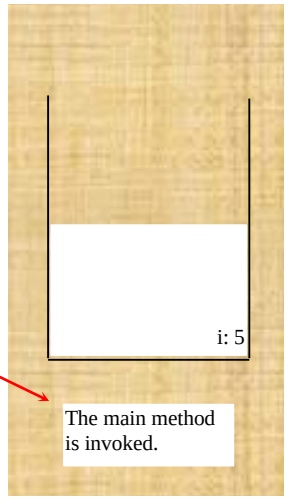
Space required for the main method
k:
j: 2
i: 5

(a) The main method is invoked.

i is declared and initialized

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

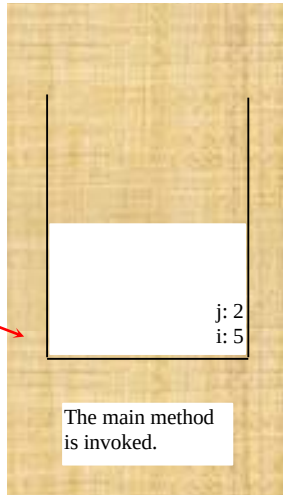
```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



j is declared and initialized

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

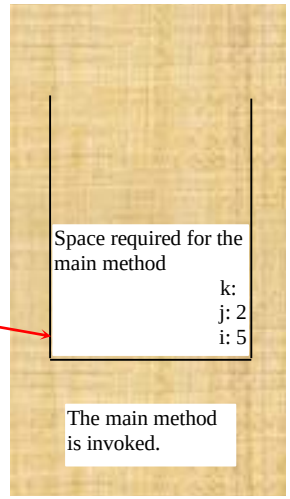
```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



Declare k

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

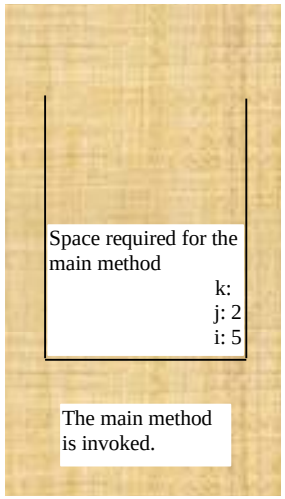
```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



Invoke max(i, j)

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

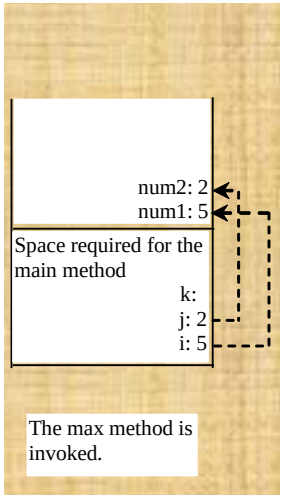
```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



pass the values of i and j to num1 and num2

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

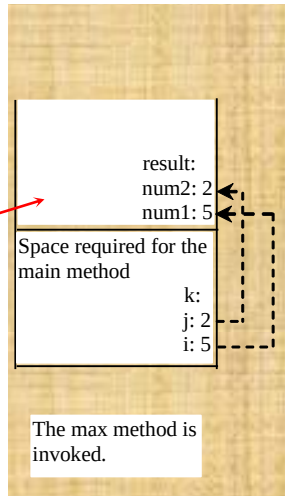
```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



pass the values of i and j to num1 and num2

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

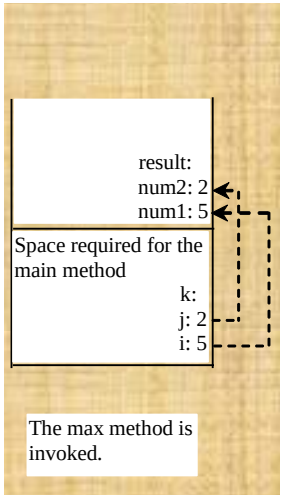
```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



(num1 > num2) is true

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

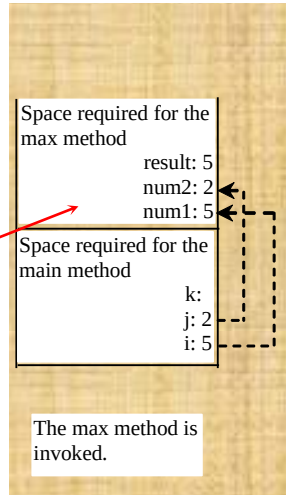
```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



Assign num1 to result

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

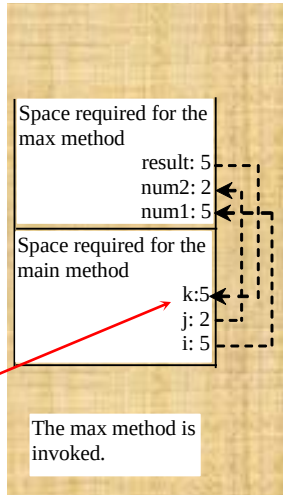
```
public static int max(int num1, int num2)
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



Return result and assign it to k

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

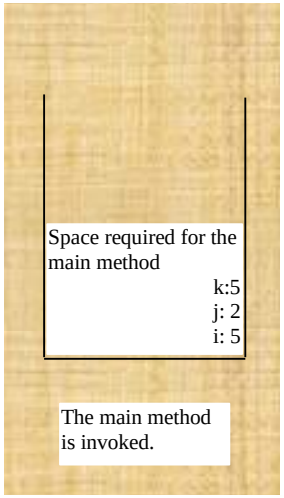
```
public static int max(int num1, int num2
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



Execute print statement

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```



- این نوع از متد، مقداری را برنمی گرداند. برخی وظایف (از جمله فراخوانی سایر متدهای برنامه) را انجام می دهد.
- شروع اجرای برنامه همواره از کلاس حاوی متد main می باشد.

مثالی از متد main

```
public class TestVoidMethod {
  public static void main(String[] args) {
    System.out.print("The grade is ");
    printGrade(78.5);

    System.out.print("The grade is ");
    printGrade(59.5);
  }

  public static void printGrade(double score) {
    if (score >= 90.0) {
      System.out.println('A');
    }
    else if (score >= 80.0) {
      System.out.println('B');
    }
    else if (score >= 70.0) {
      System.out.println('C');
    }
    else if (score >= 60.0) {
      System.out.println('D');
    }
    else {
      System.out.println('F');
    }
  }
}
```

ارسال پارامترها

```
public static void nPrintln(String message, int n) {
  for (int i = 0; i < n; i++)
    System.out.println(message);
}
```

- فرض کنید متد زیر بالا را با دستور زیر فراخوانی کرده اید:

nPrintln(“Welcome to Java”, 5);

- خروجی چیست؟

- خروجی دستور زیر چیست؟

nPrintln(“Computer Science”, 15);

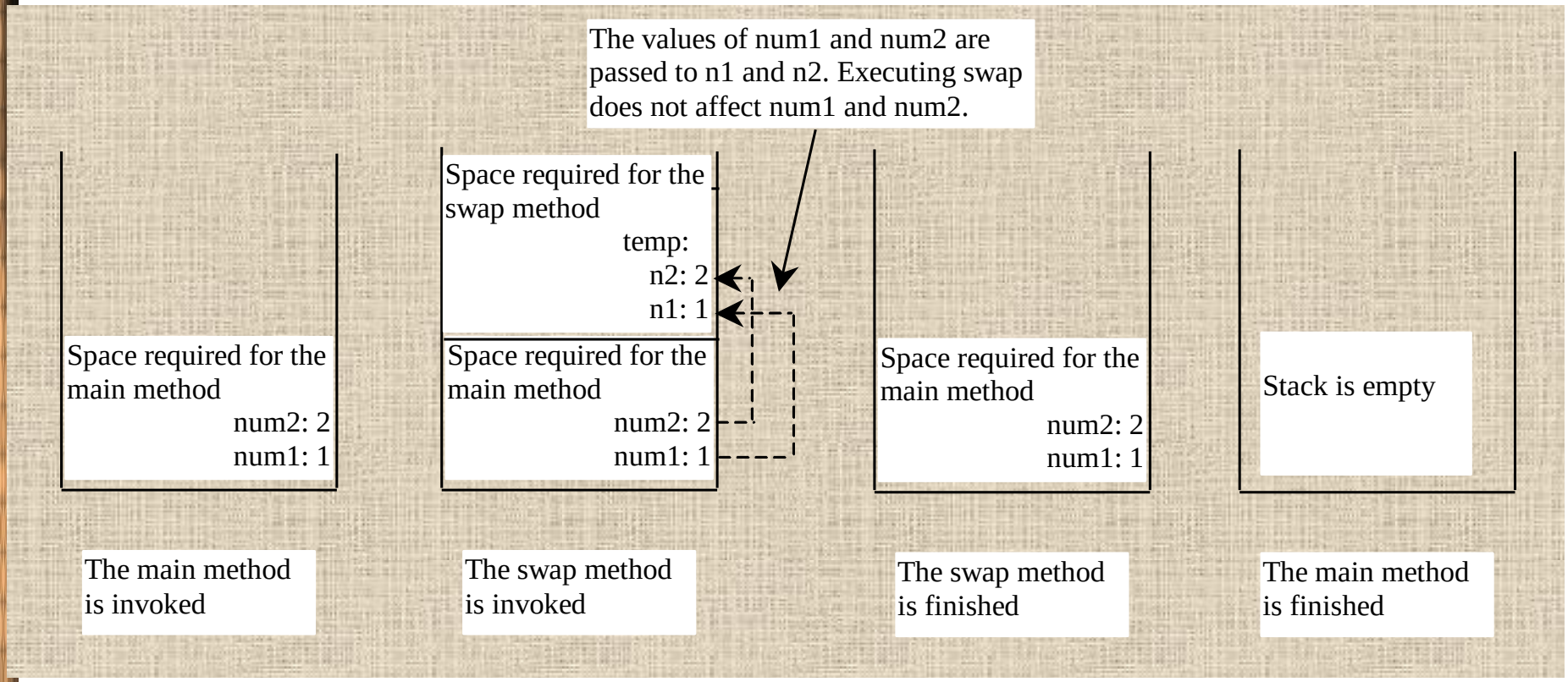
ارسال با مقدار

- برنامه نشان داده شده در اسلاید بعدی ارسال مقادیر به متدها را نمایش می دهد.

ارسال با مقدار

```
public class Increment {
  public static void main(String[] args) {
    int x = 1;
    System.out.println("Before the call, x is " + x);
    increment(x);
    System.out.println("after the call, x is " + x);
  }

  public static void increment(int n) {
    n++;
    System.out.println("n inside the method is " + n);
  }
}
```

ارسال با مقدار-ادامه



ماژوله بندی کد

- متدها می توانند به هدف کاهش عملیات تکراری در برنامه ها و استفاده مجدد از کد مورد استفاده قرار بگیرند.
- متدها همچنین می توانند برای ماژوله بندی کد و بهبود کیفیت برنامه مورد استفاده قرار گیرند.

ماژوله بندی کد-مثال بزرگترین مقسوم علیه مشترک

```
import java.util.Scanner;

public class GreatestCommonDivisorMethod {
  /** Main method */
  public static void main(String[] args) {
    // Create a Scanner
    Scanner input = new Scanner(System.in);

    // Prompt the user to enter two integers
    System.out.print("Enter first integer: ");
    int n1 = input.nextInt();
    System.out.print("Enter second integer: ");
    int n2 = input.nextInt();

    System.out.println("The greatest common divisor for " + n1 +
      " and " + n2 + " is " + gcd(n1, n2));
  }

  /** Return the gcd of two integers */
  public static int gcd(int n1, int n2) {
    int gcd = 1; // Initial gcd is 1
    int k = 1; // Possible gcd

    while (k <= n1 && k <= n2) {
      if (n1 % k == 0 && n2 % k == 0)
        gcd = k; // Update gcd
      k++;
    }

    return gcd; // Return gcd
  }
}
```

ماژوله بندی کد-مثال چاپ اعداد اول

```
public class PrimeNumberMethod {
  public static void main(String[] args) {
    System.out.println("The first 50 prime numbers are \n");
    printPrimeNumbers(50);
  }

  public static void printPrimeNumbers(int numberOfPrimes) {
    final int NUMBER_OF_PRIMES_PER_LINE = 10; // Display 10 per line
    int count = 0; // Count the number of prime numbers
    int number = 2; // A number to be tested for primeness

    // Repeatedly find prime numbers
    while (count < numberOfPrimes) {
      // Print the prime number and increase the count
      if (isPrime(number)) {
        count++; // Increase the count

        if (count % NUMBER_OF_PRIMES_PER_LINE == 0) {
          // Print the number and advance to the new line
          System.out.printf("%s\n", number);
        }
        else
          System.out.printf("%s", number);
      }

      // Check if the next number is prime
      number++;
    }
  }

  /** Check whether number is prime */
  public static boolean isPrime(int number) {
    for (int divisor = 2; divisor <= number / 2; divisor++) {
      if (number % divisor == 0) { // If true, number is not prime
        return false; // number is not a prime
      }
    }
    return true; // number is prime
  }
}
```

سربارگذاری متدها(Method Overloading)

- اگر یک کلاس دارای چندین متد همنام اما حاوی پارامترهای مختلف باشد، می گوییم سربارگذاری متد انجام شده است.
- برای عملیات یکسان، استفاده از نام یکسان برای متدهای دارای پارامترهای مختلف باعث افزایش خوانایی برنامه می شود.
- فرض کنید بخواهیم عملیات جمع را برروی دو و سه پارامتر انجام دهیم. اگر یک متد a(int,int) برای جمع دو پارامتر و یک متد b(int,int,int) برای جمع سه پارامتر داشته باشیم، هم برای برنامه نویس آن کد و هم سایر برنامه نویسانی که کد او را می خوانند، خوانایی دشوار خواهد بود.
  - زیرا از نظر خواننده، نوشتن متدهای با نامهای مختلف به مفهوم رفتار و عملکرد متفاوت آنها می باشد.
- دو روش برای سربارگذاری متد وجود دارد:
  - تغییر تعداد آرگومانها
  - تغییر نوع و ترتیب آرگومانها
- هرگاه متد سربارگذاری شده فراخوانی شود، کامپایلر جاوا متد مناسب را با بررسی تعداد، نوع و ترتیب آرگومانها انتخاب می کند.

آیا سربارگذاری متد با تغییر نوع برگشتی متد ممکن است؟

- در جاوا اینکار مجاز نمی باشد چون سبب ابهام خواهد شد.
- مثال زیر را ببینید:

```
class Calculation3{
  int sum(int a,int b){System.out.println(a+b);}
  double sum(int a,int b){System.out.println(a+b);}

  public static void main(String args[]){
  Calculation3 obj=new Calculation3();
  int result=obj.sum(20,20); //Compile Time Error

  }
}
```

Test it Now

int result=obj.sum(20,20); //Here how can java determine which sum() method should be called

آیا می توان متد main را سربارگذاری نمود؟

- بله می توان هر تعداد متد main را در یک کلاس با آرگومانهای مختلف تعریف کرد. مثال:

```
class Overloading1{
  public static void main(int a){
  System.out.println(a);
  }

  public static void main(String args[]){
  System.out.println("main() method invoked");
  main(10);
  }
}
```

Test it Now

```
Output:main() method invoked
       10
```

سربارگذاری متدها

- سربارگذاری متد max

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

سربارگذاری متدها

```
public class TestMethodOverloading {
  /** Main method */
  public static void main(String[] args) {
    // Invoke the max method with int parameters
    System.out.println("The maximum between 3 and 4 is " + max(3, 4));

    // Invoke the max method with the double parameters
    System.out.println("The maximum between 3.0 and 5.4 is " + max(3.0, 5.4));

    // Invoke the max method with three double parameters
    System.out.println("The maximum between 3.0, 5.4, and 10.14 is " + max(3.0, 5.4, 10.14));
  }

  /** Return the max between two int values */
  public static double max(int num1, int num2) {
    if (num1 > num2)
      return num1;
    else
      return num2;
  }

  /** Find the max between two double values */
  public static double max(double num1, double num2) {
    if (num1 > num2)
      return num1;
    else
      return num2;
  }
  /** Return the max among three double values */
  public static double max(double num1, double num2, double num3) {
    return max(max(num1, num2), num3);
  }
}
```

فراخوانی مبهم

```
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;
  }
}
```

متد تبدیل عدد مبنای 10 به مبنای 16

- متدی بنویسید که یک عدد صحیح مبنای 10 را به مبنای 16 تبدیل نماید.

حوزه متغیر های محلی

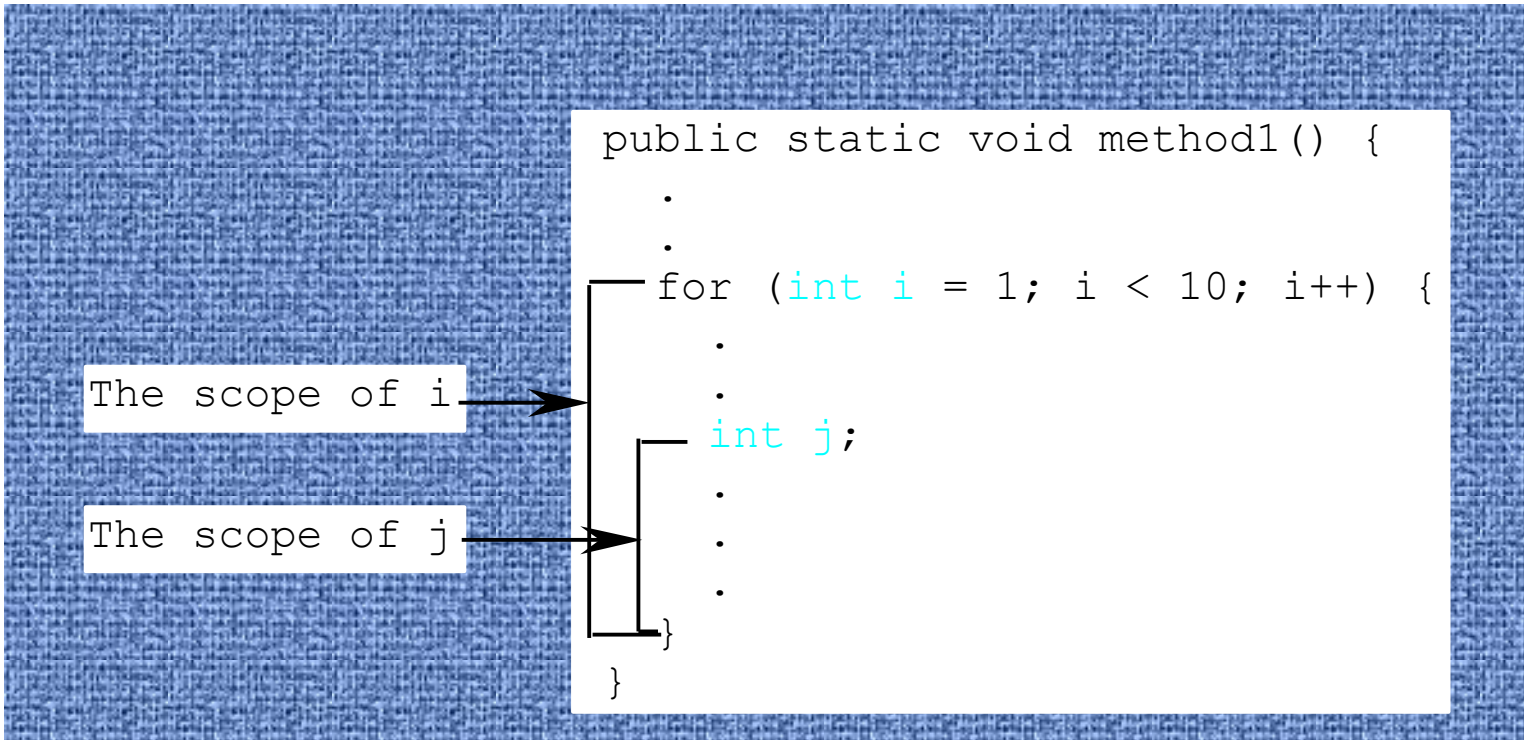
- یک متغیر محلی: متغیری است که درون یک متد تعریف شده است.
- حوزه (Scope) : بخشی از برنامه که متغیر می تواند در آنجا مورد استفاده قرار گیرد.
- حوزه یک متغیرمحلی از نقطه اعلان (تعریف) آن آغاز شده و تا پایان بلاکی است که متغیر را در برگرفته است.
- یک متغیر محلی باید پیش از استفاده، حتماً اعلان شده باشد.

حوزه متغیر های محلی-ادامه

- شما می توانید یک متغیر محلی را با همان نام بارها در بلاکهای مختلف غیرتودرتو تعریف کنید، اما در بلاکهای تودرتو یک متغیر نباید بیش از یک بار تعریف شود.

حوزه متغیر های محلی-ادامه

- متغیری که در بخش کنترلی یک حلقه for تعریف شده است در کل بدنه حلقه دارای حوزه قلمرو است. اما متغیری که درون بدنه حلقه تعریف شده دارای حوزه ای از شروع محل تعریف تا پایان بلاکی است که آن را دربرگرفته است.



حوزه متغیر های محلی-ادامه

It is fine to declare i in two non-nesting blocks

```
public static void method1() {
  int x = 1;
  int y = 1;

  for (int i = 1; i < 10; i++) {
    x += i;
  }

  for (int i = 1; i < 10; i++) {
    y += i;
  }
}
```

It is wrong to declare i in two nesting blocks

```
public static void method2() {

  int i = 1;
  int sum = 0;

  for (int i = 1; i < 10; i++)
    sum += i;
  }
}
```

حوزه متغیر های محلی-ادامه

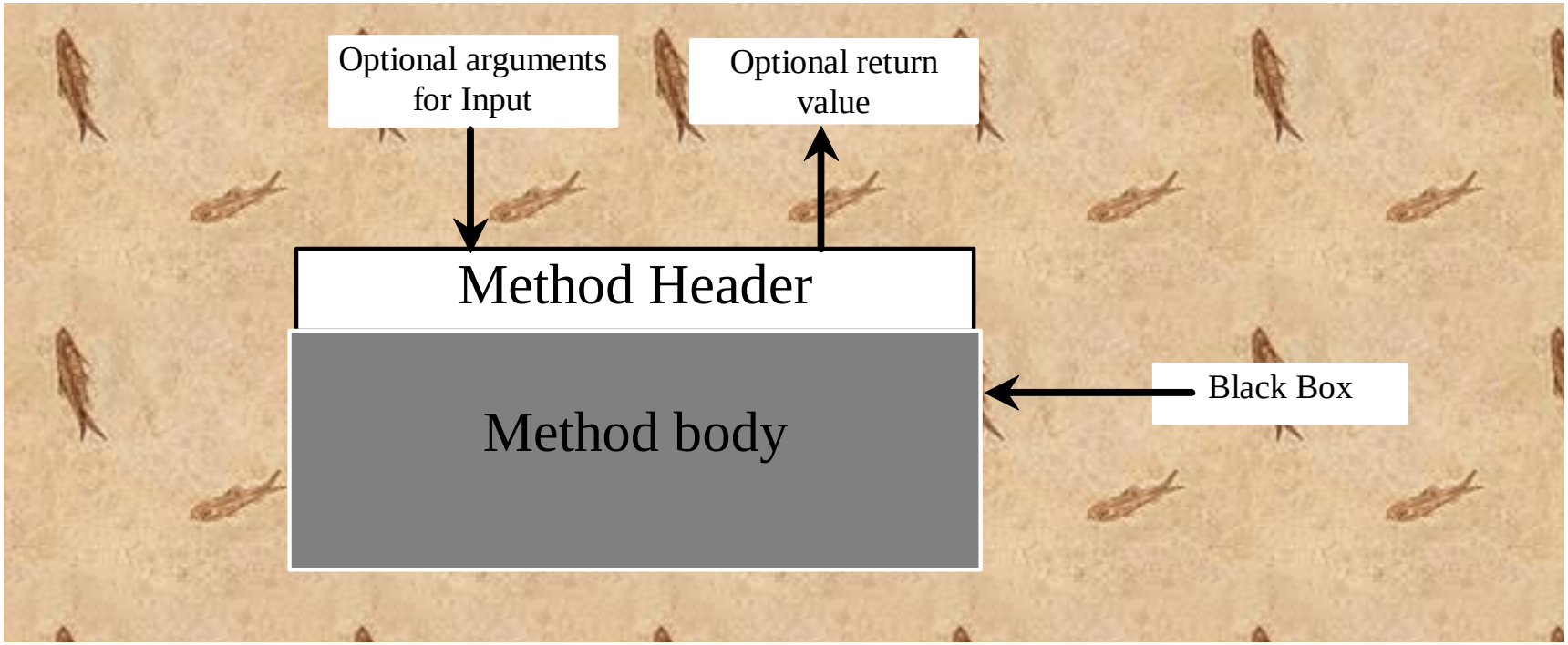
```
// Fine with no errors
public static void correctMethod() {
   int x = 1;
   int y = 1;
   // i is declared
   for (int i = 1; i < 10; i++) {
      x += i;
   }
   // i is declared again
   for (int i = 1; i < 10; i++) {
      y += i;
   }
}
```

حوزه متغیر های محلی-ادامه

```
// With errors
public static void incorrectMethod() {
   int x = 1;
   int y = 1;
   for (int i = 1; i < 10; i++) {
     int x = 0;
     x += i;
   }
}
```

انتزاع متد (method abstraction)

- شما می توانید بدنه یک متد را به مانند یک جعبه سیاه درنظر بگیرید که حاوی جزئیات پیاده سازی آن متد است.



متدهای مثلثاتی

- sin(double a)
- cos(double a)
- tan(double a)
- acos(double a)
- asin(double a)
- atan(double a)

مثالها:

```
Math.sin(0) returns 0.0
Math.sin(Math.PI / 6)
  returns 0.5
Math.sin(Math.PI / 2)
  returns 1.0
Math.cos(0) returns 1.0
Math.cos(Math.PI / 6)
  returns 0.866
Math.cos(Math.PI / 2)
  returns 0
```

متدهای توان(نما)

- `exp(double a)`
  - e به توان a را بر می گرداند.
- `log(double a)`
  - لگاریتم طبیعی a را بر می گرداند.
- `log10(double a)`
  - لگاریتم مبنای 10 مقدار a را بر می گرداند.
- `pow(double a, double b)`
  - a به توان b را بر می گرداند.
- `sqrt(double a)`
  - ریشه دوم a را بر می گرداند.

مثالها:

```
Math.exp(1) returns 2.71
Math.log(2.71) returns 1.0
Math.pow(2, 3) returns 8.0
Math.pow(3, 2) returns 9.0
Math.pow(3.5, 2.5) returns
   22.91765
Math.sqrt(4) returns 2.0
Math.sqrt(10.5) returns 3.24
```

متدهای روندسازی

- `double ceil(double x)`

  - X به نزدیکترین مقدار صحیح بزرگتر از خود روند می شود. این مقدار صحیح در قالب یک double برگردانده می شود.

- `double floor(double x)`

  -X به نزدیکترین مقدار صحیح کوچکتر از خود روند می شود. این مقدار صحیح در قالب یک double برگردانده می شود.

- `double rint(double x)`

  -X به نزدیکترین مقدار صحیح روند می شود. اگر x به دو عدد صحیح فاصله یکسانی داشته باشد، مقداری که زوج است در قالب doubleبرگردانده می شود.

- `int round(float x)`

  Return (int)Math.floor(x+0.5).

- `long round(double x)`

  Return (long)Math.floor(x+0.5).

متدهای روندسازی-مثالها

```
Math.ceil(2.1)
Math.ceil(2.0)
Math.ceil(-2.0)
Math.ceil(-2.1)
Math.floor(2.1)
Math.floor(2.0)
Math.floor(-2.0)
Math.floor(-2.1)
Math.rint(2.1)
Math.rint(2.0)
Math.rint(-2.0)
Math.rint(-2.1)
Math.rint(2.5)
Math.rint(-2.5)
Math.round(2.6f)
Math.round(2.0)
Math.round(-2.0f)
Math.round(-2.6)
```

متدهای min, max, abs

- `max(a, b)` and `min(a, b)`
  - مقدار ماکزیمم یا مینیمم دوپارامتر ورودی را بر می گرداند.
- `abs(a)`
  - مقدار قدرمطلق a را بر می گرداند.
- `random()`
  - یک مقدار تصادفی از نوع double در بازه [0.0, 1.0) تولید می کند.

```
Examples:


Math.max(2, 3) returns 3
Math.max(2.5, 3) returns
  3.0
Math.min(2.5, 3.6)
  returns 2.5
Math.abs(-2) returns 2
Math.abs(-2.1) returns
  2.1
```

متد رندوم

مثالها:

```
(int)(Math.random() * 10)
```
→ Returns a random integer between 0 and 9.

```
50 + (int)(Math.random() * 50)
```
→ Returns a random integer between 50 and 99.

در حالت کلی:

```
a + Math.random() * b
```
→ Returns a random number between a and a + b, excluding a + b.