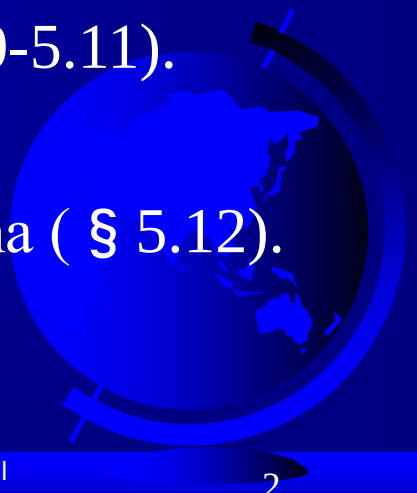
Bölüm 5: Metotlar

Amaçlar

- Metot tanımlamak, metot çağırmak ve bir metoda argümanları iletmek (§ 5.2-5.5).
- Modüler, okunması, hata ayıklanması ve bakımı kolay yeniden kullanılabilir kod geliştirme(§ 5.6).
- Aşırı yüklenmiş metot (method overloading) kullanmak ve çok anlamlı aşırı yüklemeyi anlamak (§ 5.7).
- Aşırı yüklenmiş metot tasarlanması ve uygulaması (§ 5.8).
- Değişkenlerin kapsamını belirlemek(§ 5.9).
- Math sınıfının içinde metotlar kullanma (§ § 5.10-5.11).
- Metot soyutlama kavramını öğrenme (§ 5.12).
- Kademeli geliştirilmiş metot tasarlama ve kullanma (§ 5.12).



Açılış Problemi

Sırasıyla, 1'den 10' a kadar, 20'den 30'a kadar ve 35'ten 45'e kadar olan sayıların toplamını bulunuz

Problem

```
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);
```

Problem

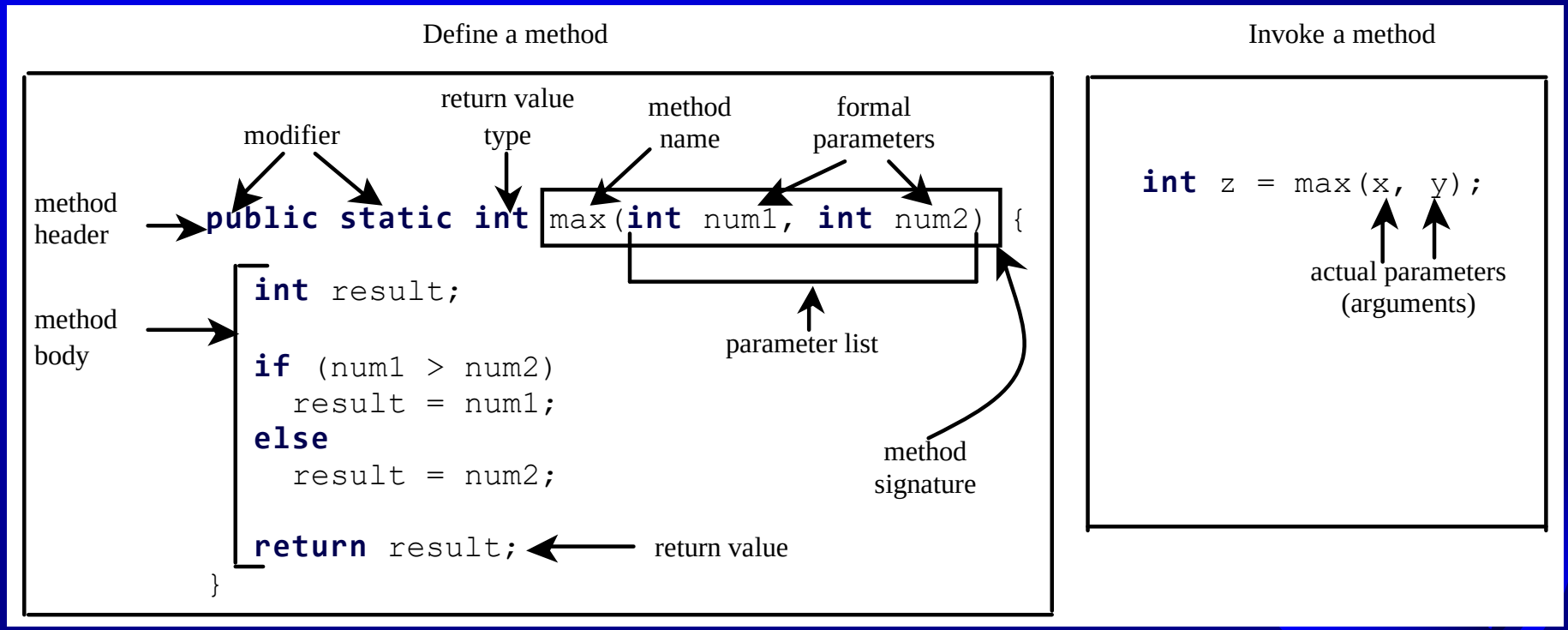
```
int sum = 0;
for (int i = 1; i <= 10; i++)
  sum += i;
System.out.println("Sum from 1 to 10 is " + sum);

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);
```

Çözüm

```
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));
}
```

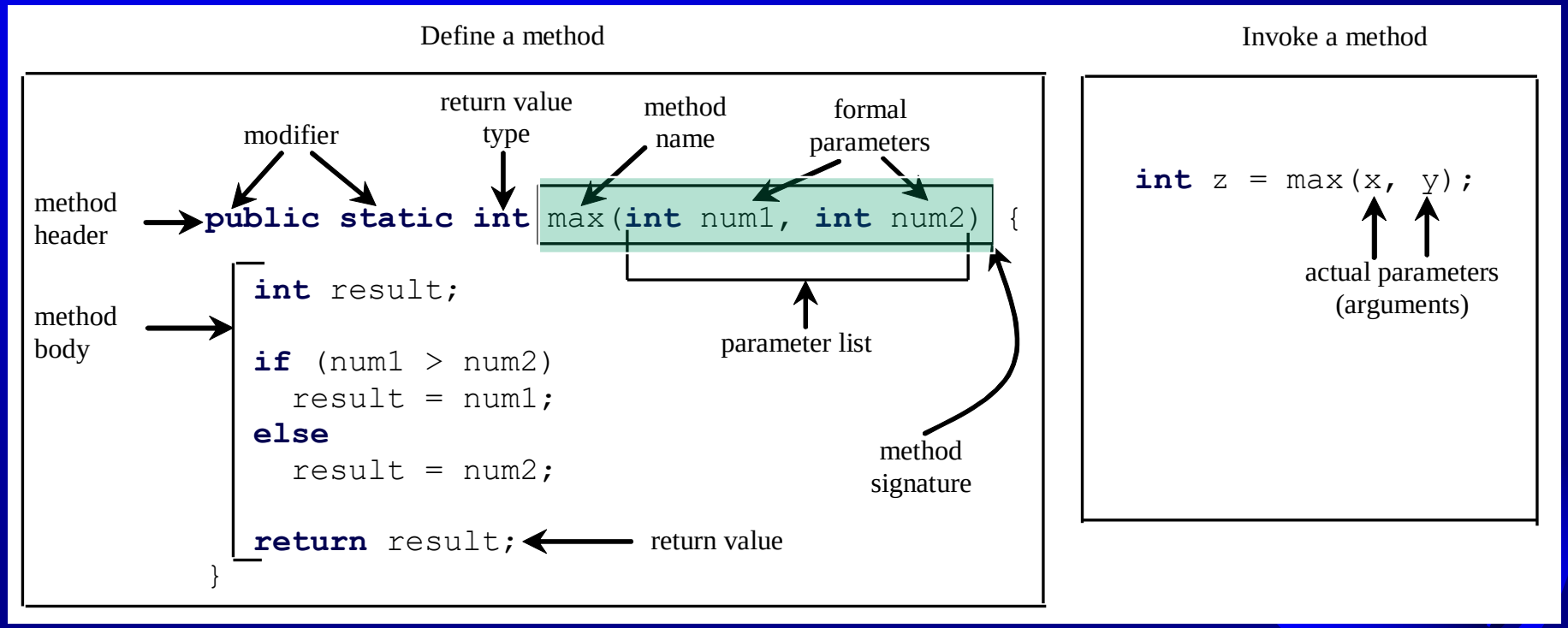
Metot tanımı

METOT, bir işlemi gerçekleştirmek için birlikte gruplanmış ifadeler topluluğudur.



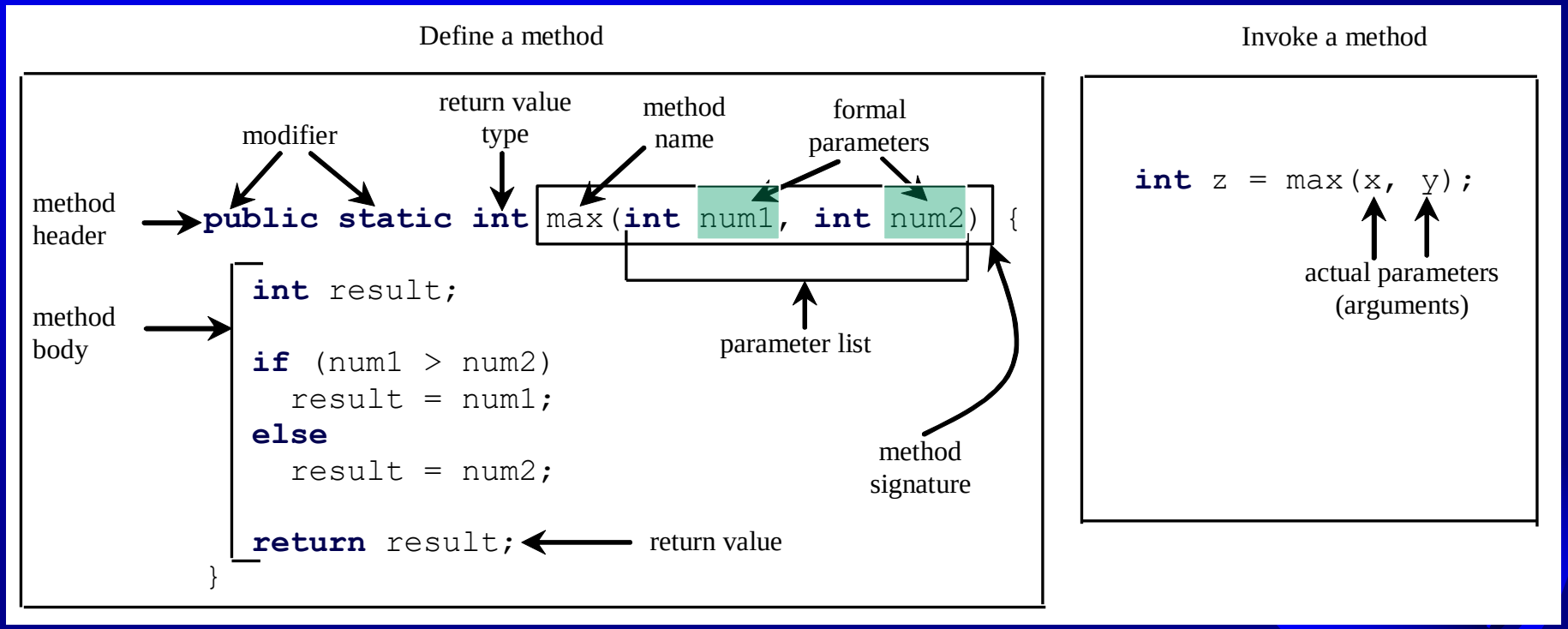
Metot imzası (Method Signature)

Metot imzası, metot adı ile parametrelerinin birleşimidir.



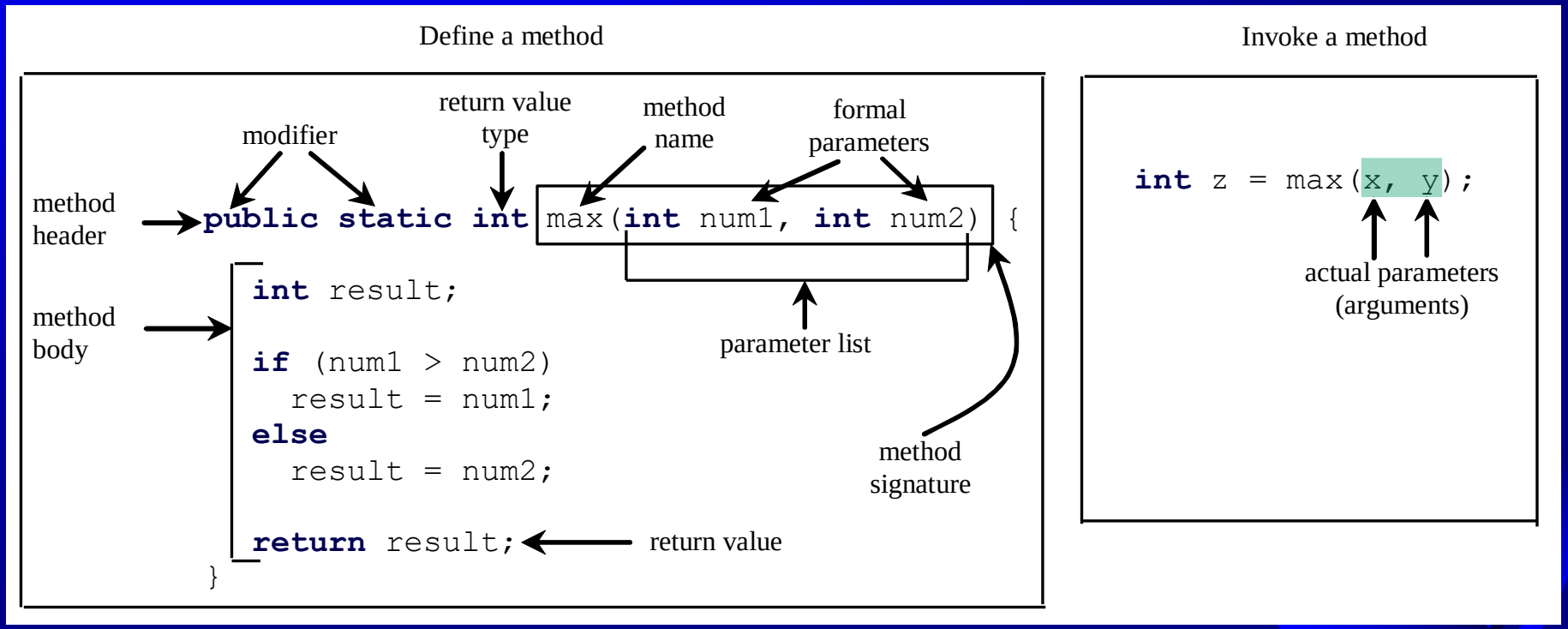
Formal Parametreler

Metot başlığı içinde tanımlanmış değişkenlere formal parametreler denir.



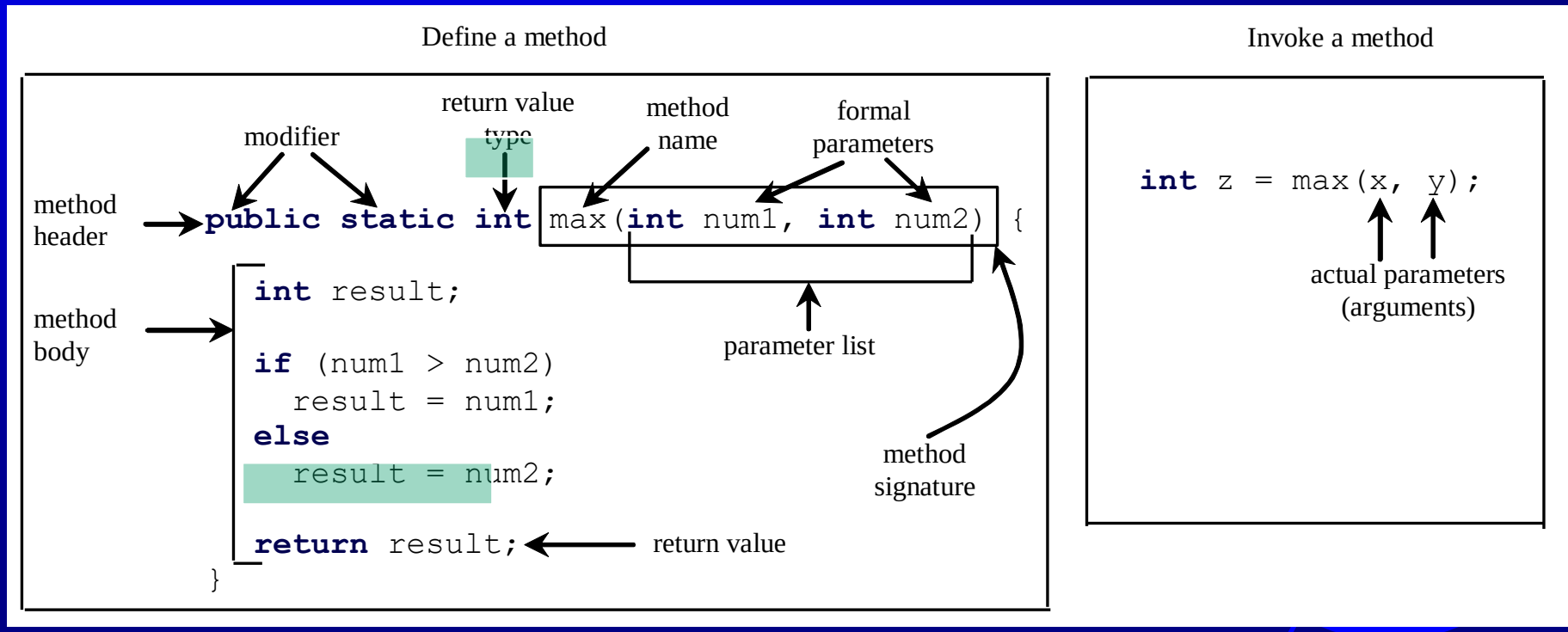
Gerçek Parametreler

Bir metot çağrıldığında, bir değer parametreye geçer . Bu değere gerçek parametre veya argüman denir (*actual parameter or argument*).



Geridönüş değeri tipi (return value type)

Bir metotbir değer döndürebilir. returnValueType, metodun döndürdüğü değerin veri tipidir. Eğer metot bir değer döndürmüyorsa, returnValueType için void kullanılır. Örneğin main metodunun geridönüş değerinin tipi void türündedir.



Metotların çağırılması

`max` metodunu test edelim

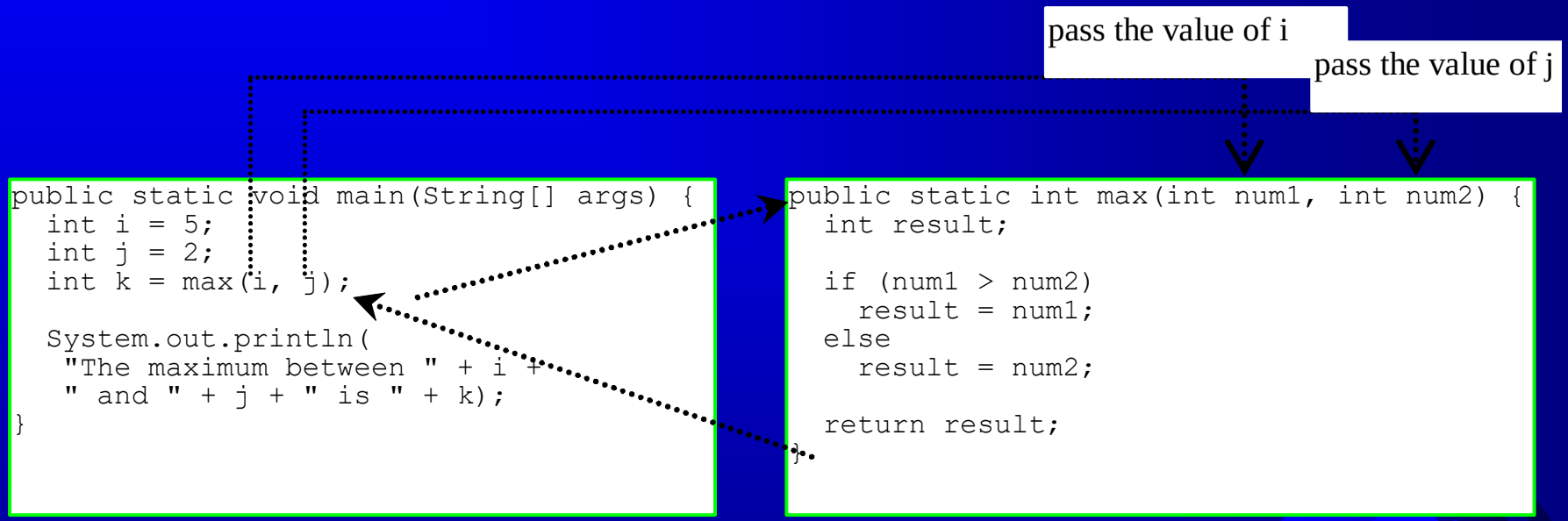
Bu program int tipindeki değerlerin en büyüğünü döndüren max metodunu çağırır.

TestMax

Run

animation

Metotların çağrılması



animation

Metot çağrılmasının yürütülmesi

i 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;
}
```

animation

Metod çağrılmasının yürütülmesi

j is now 2

```
public static void main(Strin[] 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;
}
```

animation

Metot çağrılmasının yürütülmesi

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;
}
```

animation

Metot çağrılmasının yürütülmesi

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;
}
```

animation

Metot çağrılmasının yürütülmesi

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;
}
```

animation

Metot çağrılmasının yürütülmesi

(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;
}
```

animation

Metot çağrılmasının yürütülmesi

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;
}
```

animation

Metot çağrılmasının yürütülmesi

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;
}
```

animation

Metot çağrılmasının yürütülmesi

return max(i, j) and assign the return value 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;
}
```

animation

Metot çağrılmasının yürütülmesi

Execute the 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;
}
```

DİKKAT

Değer döndüren metotlar için return ifadesi gereklidir. (a) da gösterilen metot mantıksal olarak doğru fakat derleme hatası içerir. Çünkü Java derleyicisi bu metodun herhangi bir değer döndürmemesinin mümkün olduğunu düşünüyor.

```
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)

Hatayı düzeltmek için, *if (n < 0)* ifadesini (a)'dan silin, böylece derleyici if ifadesinin nasıl değerlendirildiğine bakmaksızın return ifadesini görecektir.

Diğer sınıflarda yeniden kullanılabilen metotlar

NOT: Metotların faydalarından biri de yeniden kullanım özelliklerinin olmasıdır. max metodu, TestMax sınıfının yanısıra herhangi bir sınıftan da çağırılabilir . Eğer Test adında yeni bir sınıf oluşturursanız max metodunu ClassName.methodName kullanarak bu sınıf içerisinden de çağırabilirsiniz. (e.g., TestMax.max).

void Metot Örneği

Void tipindeki metotler değer döndürmez sadece bir takım eylemleri gerçekleştirir.

TestVoidMethod

Run

Parametre Geçişi (Passing Parameters)

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

Aşağıdaki metot çağrıldığında çıktı ne olur?
nPrintln(“Welcome to Java”, 5);

Aşağıdaki metot çağrıldığında çıktı ne olur?
nPrintln(“Computer Science”, 15);

Değer Geçişi (Pass by Value)

Bu program bir metoda değer geçişini gösterir.

Değer Geçişi (Pass by Value)

Değer geçişini test etme (Testing Pass by value)

Bu program bir metoda değer geçişini gösterir.

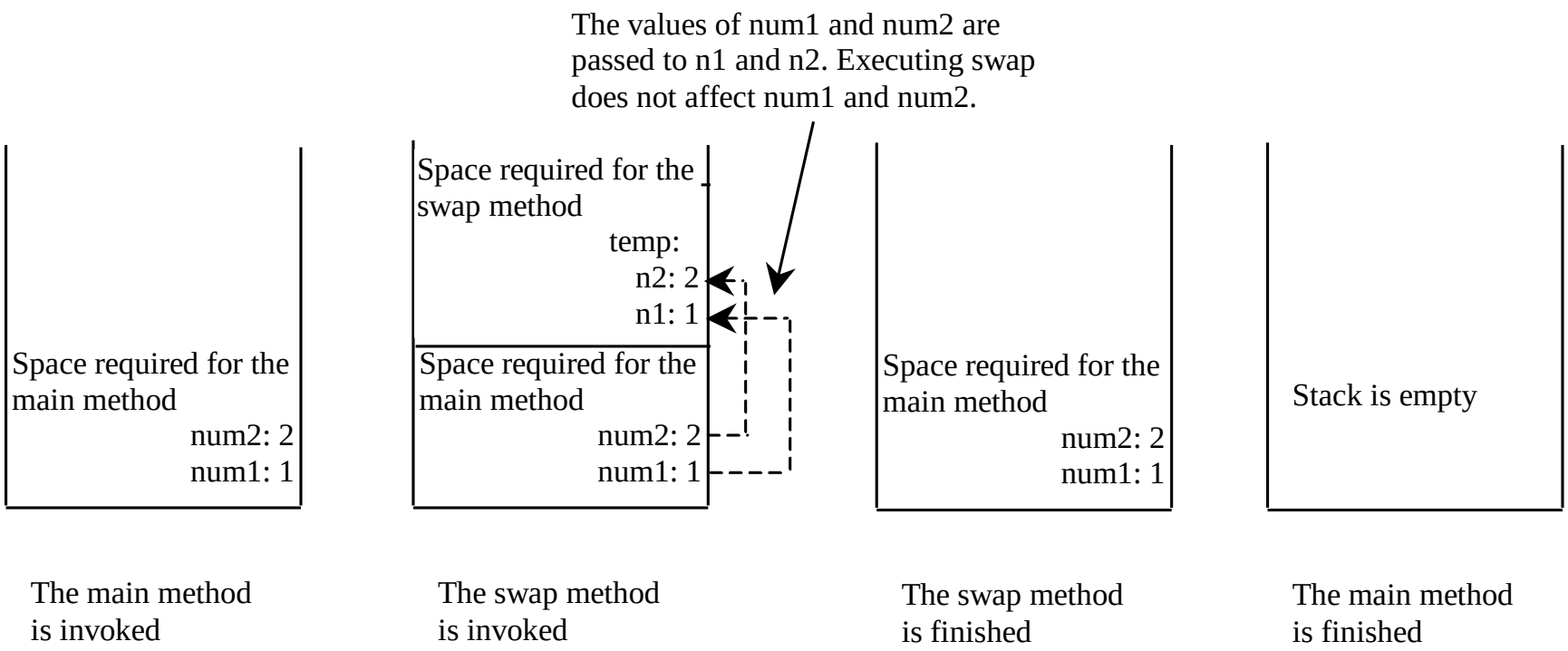
TestPassByValue

Run

LISTING 6.5 TestPassByValue.java

```
1  public class TestPassByValue {
2    /** Main method */
3    public static void main(String[] args) {
4      // Declare and initialize variables
5      int num1 = 1;
6      int num2 = 2;
7
8      System.out.println("Before invoking the swap method, num1 is " +
9        num1 + " and num2 is " + num2);
10
11     // Invoke the swap method to attempt to swap two variables
12     swap(num1, num2);
13
14     System.out.println("After invoking the swap method, num1 is " +
15       num1 + " and num2 is " + num2);
16   }
17
18   /** Swap two variables */
19   public static void swap(int n1, int n2) {
20     System.out.println("\tInside the swap method");
21     System.out.println("\t\tBefore swapping, n1 is " + n1
22       + " and n2 is " + n2);
23
24     // Swap n1 with n2
25     int temp = n1;
26     n1 = n2;
27     n2 = temp;
28
29     System.out.println("\t\tAfter swapping, n1 is " + n1
30       + " and n2 is " + n2);
31   }
32 }
```

Değer Geçişi (Pass by Value, cont.)



Modüler Kod yazımı

Metotlar gereksiz kod yazımını azaltmak ve kodun yeniden kullanılabilirliği sağlamak amacıyla kullanılabilir. Metotlar ayrıca, kodu modüllere ayırmak ve programın kalitesini geliştirmek amacıyla da kullanılabilir.

GreatestCommonDivisorMethod

Run

Aşırı yüklenmiş metodlar (Overloading Methods)

Overloading metodlar aynı metot adı ile farklı parametrelere sahip metotlar yazmayı sağlar .

```
public static double max(double num1, double num2) // max(7.5 , 4.8)

public static int max(int num1, int num2) //
max(3 , 5)
```

Aşırı yüklenmiş metodlar (Overloading Methods)

Aşırı yüklenmiş `max` Metodu

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

TestMethodOverloading

Run

Belirsiz Çağırma (Ambiguous Invocation)

Bazen bir metod çağrıldığında, iki veya daha fazla eşleşme olabilir. Derleyici hangisi ile eşleştireceğine karar veremez Bu durum belirsiz çağırma olarak adlandırılır ve derleme hatasıdır.

Ambiguous Invocation

```
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;
  }
}
```

Yerel değişkenlerin kapsamı

Yerel değişken: metodun içinde tanımlanan değişkendir.

Kapsamı: programda değişkenin referans alınabileceği.

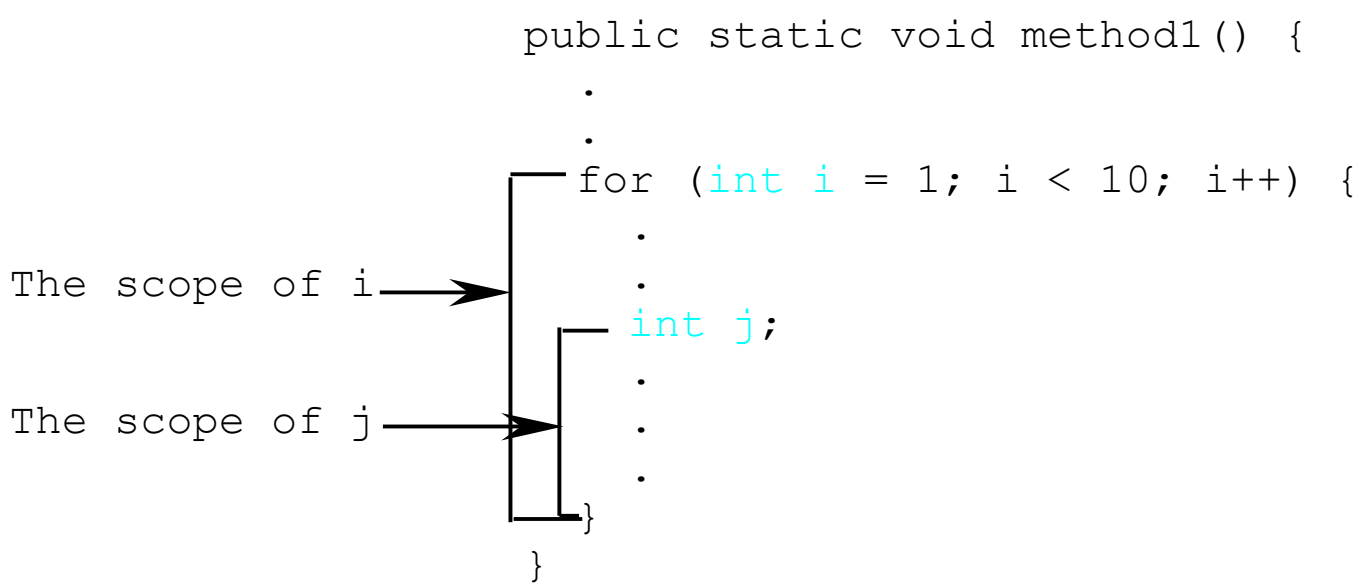
Yerel değişkenin kapsamı, bidirildiği yerde başlar ve değişkeni içeren bloğun sonuna kadar devam eder. Yerel değişken kullanılmadan önce bildirilmelidir (declared).

Yerel değişkenlerin kapsamı

Aynı adda bir yerel değişken, bir metoddaki iç içe geçmeyen bloklar içerisinde birçok kez tanımlanabilir. Fakat iç içe geçmiş (nested) bloklarda iki kez tanımlanamaz.

Yerel değişkenlerin kapsamı

For döngüsü içerisinde tanımlanan değişkenin tüm döngü içerisinde geçerlidir. Fakat for döngüsünün gövdesinde tanımlanan bir değişken, değişkenin ilk tanımlanmasından değişkenin tanımlandığı bloğun sonuna kadar geçerlidir..



Yerel değişkenlerin kapsamı

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;
  }
}
```

Yerel değişkenlerin kapsamı

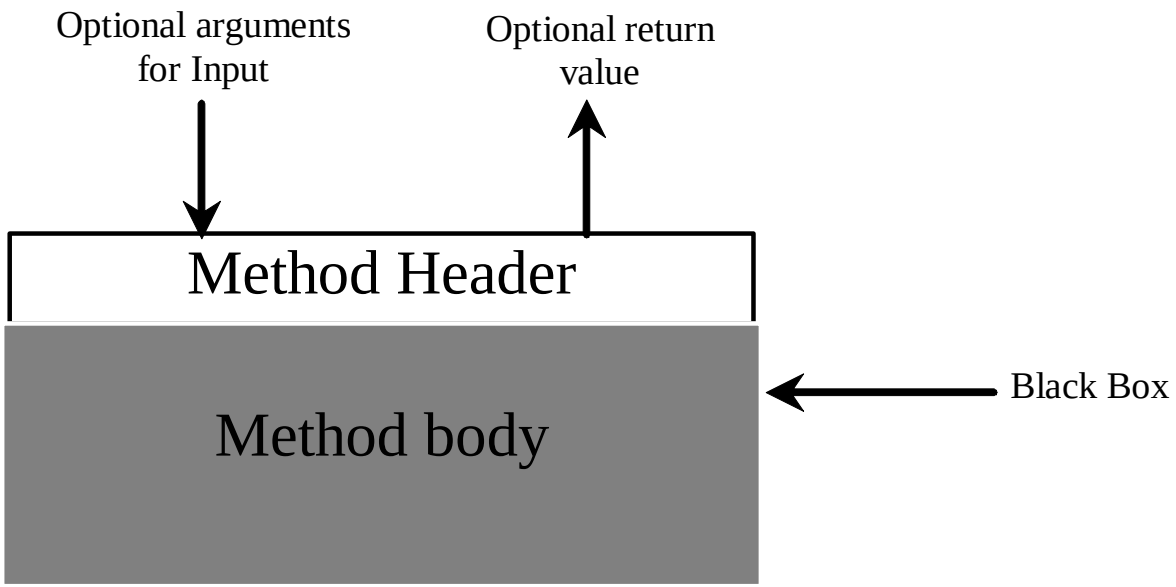
```
// Fine with no errors
public static void correctMethod() {
  int x = 1;
  int y = 1;
  // i is declared
  for (int i = 1; i < 10; i++) {
    x += i;
  }
  // i is declared again
  for (int i = 1; i < 10; i++) {
    y += i;
  }
}
```

Yerel değişkenlerin kapsamı

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

Metot Soyutlama

Metot gövdesini, yöntemin ayrıntılarını içeren bir kara kutu olarak düşünebilirsiniz.



Metotların Faydası

- Metodu bir kez yaz ve her yerde kullan
- Bilgi saklama: Metodun uygulama ayrıntısını kullanıcıdan gizleme.
- Karmaşıklığı azaltma

`Math` Sınıfı

- Sınıf sabitleri:
  - `PI`
  - `E`
- Sınıfın metotları:
  - Trigonometrik Metotlar
  - Exponent Metotlar
  - Yuvarlama Metotları
  - min, max, abs, ve random Metotları

Trigonometrik Metotlar

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

Radians

toRadians(90)

```
Examples:

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
```

Exponent (Üstel) Metotlar

- `exp(double a)`
  Returns `e` raised to the power of `a`.
- `log(double a)`
  Returns the natural logarithm of `a`.
- `log10(double a)`
  Returns the 10-based logarithm of `a`.
- `pow(double a, double b)`
  Returns `a` raised to the power of `b`.
- `sqrt(double a)`
  Returns the square root of `a`.

```
Examples:

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
```

Yuvarlama (Rounding) Metotları

- `double ceil(double x)`

  x rounded up to its nearest integer. This integer is returned as a double value.

- `double floor(double x)`

  x is rounded down to its nearest integer. This integer is returned as a double value.

- `double rint(double x)`

  x is rounded to its nearest integer. If x is equally close to two integers, the even one is returned as a double.

- `int round(float x)`

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

- `long round(double x)`

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

Yuvarlama Metodları Örnekleri

```
Math.ceil(2.1) returns 3.0
Math.ceil(2.0) returns 2.0
Math.ceil(-2.0) returns -2.0
Math.ceil(-2.1) returns -2.0
Math.floor(2.1) returns 2.0
Math.floor(2.0) returns 2.0
Math.floor(-2.0) returns -2.0
Math.floor(-2.1) returns -3.0
Math.rint(2.1) returns 2.0
Math.rint(2.0) returns 2.0
Math.rint(-2.0) returns -2.0
Math.rint(-2.1) returns -2.0
Math.rint(2.5) returns 2.0
Math.rint(-2.5) returns -2.0
Math.round(2.6f) returns 3
Math.round(2.0) returns 2
Math.round(-2.0f) returns -2
Math.round(-2.6) returns -3
```

min, max, ve abs

- `max(a, b)` and `min(a, b)`

  Returns the maximum or minimum of two parameters.

- `abs(a)`

  Returns the absolute value of the parameter.

- `random()`

  Returns a random `double` value in the range [0.0, 1.0).

```
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
```

random Metodu

0.0 ile 1.0 arasında 0 dahil rastgele double tipinde sayı üretir.

(0 <= Math.random() < 1.0).

Örnekler:

```
(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.
```

Genelde,

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

Örnek Çalışma:Rastgele karakter üretme

Bilgisayar programları sayısal verileri ve karakterleri işler. Bu ana kadar sayısal veri içeren bir çok örnek gördük. Bunun yanısıra karakterleri ve bu veri tipini nasıl işleyeceğimiz de önemlidir.

2.9 da anlatıldığı gibi, her karakterin kendine has 0 ve FFFF arasında hexadecimal sistemde Unicode'u vardır. (65535 desimal sitemde). 0 ve 65535 arasında rastgele karakter üretmek için şu ifade kullanılır. (Not: 0 <= Math.random() < 1.0 formülünü 1 ile 65535 arasına çekmek gerek.)

```
(int)(Math.random() * (65535 + 1))
```

Örnek Çalışma: Rastgele karakter üretme

Şimdi nasıl rastgele küçük harf üretildiğini görelim. Küçük harfllerin Unicode'ları ardışık tam sayılar olup 'a', 'b', 'c', ..., ve 'z' şeklindedir.

'a' için Unicode:

(int)'a'

Dolayısıyla (int)'a' ile (int)'z' arasında rastgele sayı üretmek demek herhangi bir karakter üretmek anlamına gelir:

(int)((int)'a' + Math.random() * ((int)'z' - (int)'a' + 1)

Örnek Çalışma:Rastgele karakter üretme

Bölüm 2'de tüm nümerik operatörlerin karakter operandlarına uygulanabileceğini görmüştük. Char operandı sayıya dönüştürülebilir (cast) Eğer diğer operand sayı veya karakter ise karakter bir sayıya dönüştürülebilir. Bu yüzden, önceki ifade aşağıdaki gibi sadeleştirilebilir:

```
'a' + Math.random() * ('z' - 'a' + 1)
```

Rastgele küçük harf aşağıdaki gibi üretilebilir:

```
(char)('a' + Math.random() * ('z' - 'a' + 1))
```

Örnek Çalışma:Rastgele karakter üretme

Genel yapı elde etmek adına, ch1 ve ch2 gibi iki karakter arasında rastgele bir karakter üretimi (ch1 < ch2 olmak koşulu ile aşağıdaki gibi yazılabilir:

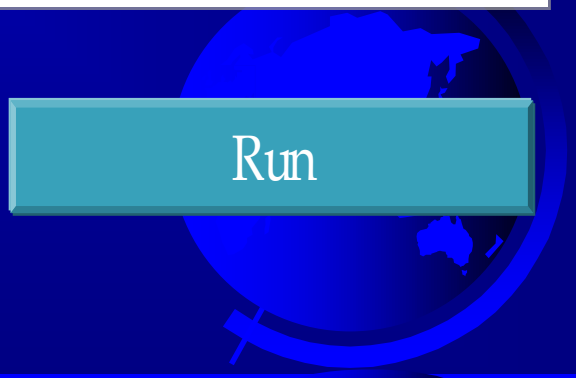
(char)(ch1 + Math.random() * (ch2 – ch1 + 1))

RandomCharacter Sınıfı

```
// RandomCharacter.java: Generate random characters
public class RandomCharacter {
  /** Generate a random character between ch1 and ch2 */
  public static char getRandomCharacter(char ch1, char ch2) {
    return (char)(ch1 + Math.random() * (ch2 - ch1 + 1));
  }

  /** Generate a random lowercase letter */
  public static char getRandomLowerCaseLetter() {
    return getRandomCharacter('a', 'z');
  }

  /** Generate a random uppercase letter */
  public static char getRandomUpperCaseLetter() {
    return getRandomCharacter('A', 'Z');
  }

  /** Generate a random digit character */
  public static char getRandomDigitCharacter() {
    return getRandomCharacter('0', '9');
  }

  /** Generate a random character */
  public static char getRandomCharacter() {
    return getRandomCharacter('\u0000', '\uFFFF');
  }
}
```

RandomCharacter

TestRandomCharacter

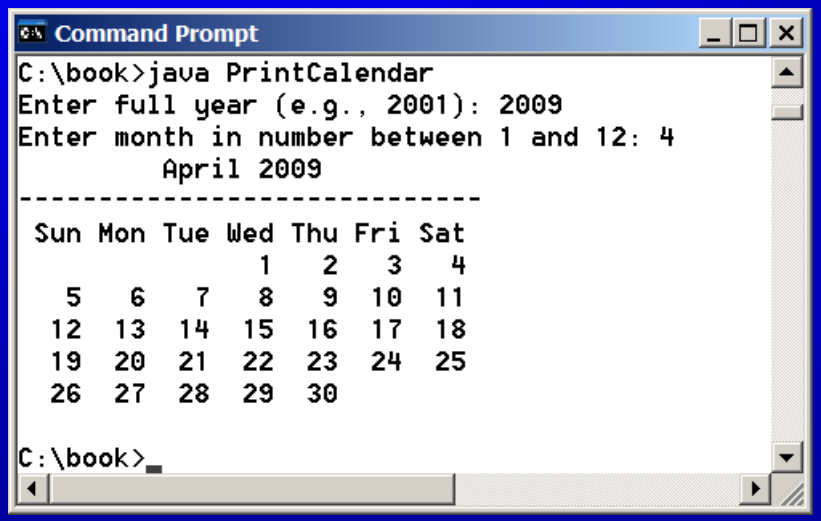


Kademeli geliştirme (Stepwise refinement)

Metot soyutlama kavramı, program geliştirme sürecine uygulanabilir. Büyük bir program yazarken, onu kademeli geliştirme olarak da bilinen, alt problemlere ayırma mantığı ile çalışan "böl ve yönet" stratejisini kullanabilirsiniz. Alt problemler daha küçük, daha yönetilebilir problemlere ayrıştırılabilir

PrintCalender Örneği

Kademeli geliştirmeyi gösterebilmek adına Takvim yazma (PrintCalendar) örneğini ele alabiliriz.



PrintCalendar

Run

Diagram Tasarımı

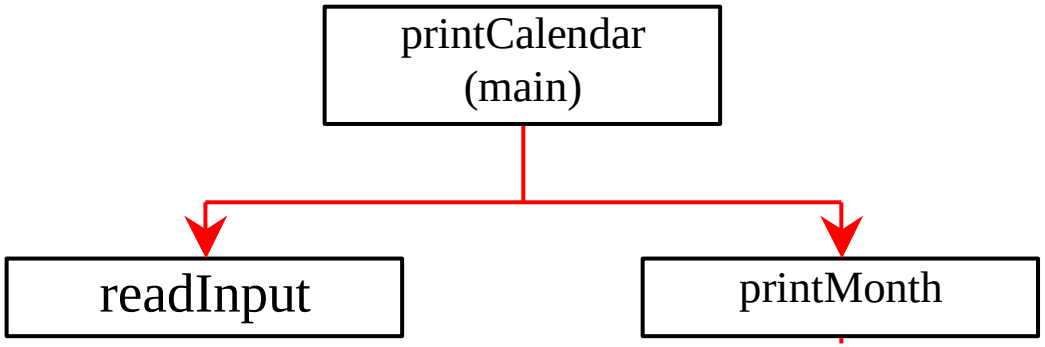


Diagram Tasarımı

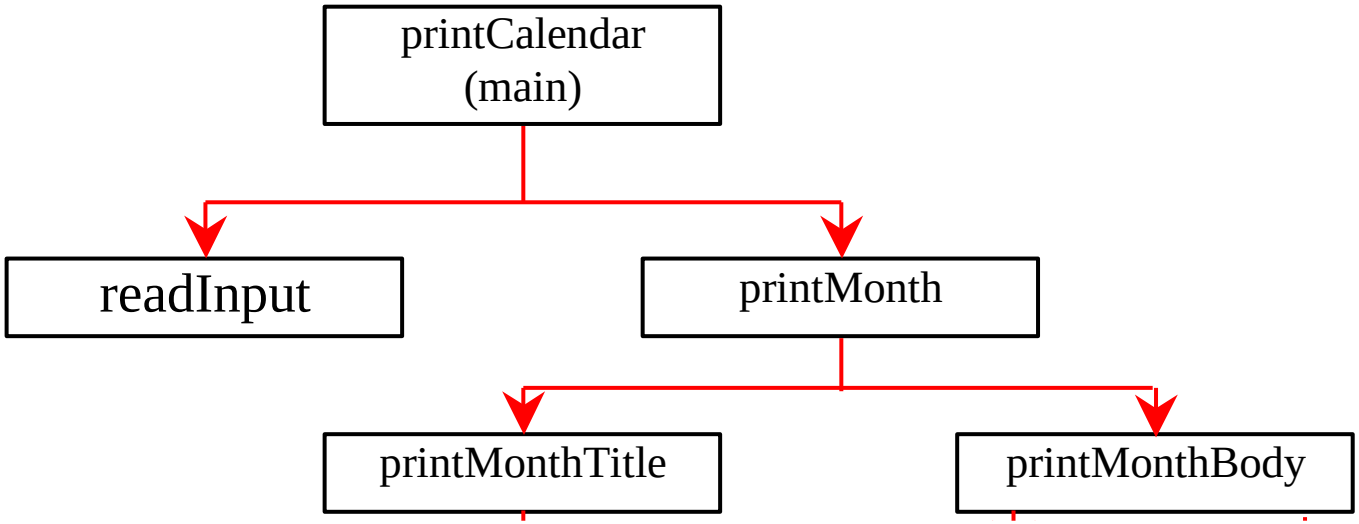


Diagram Tasarımı

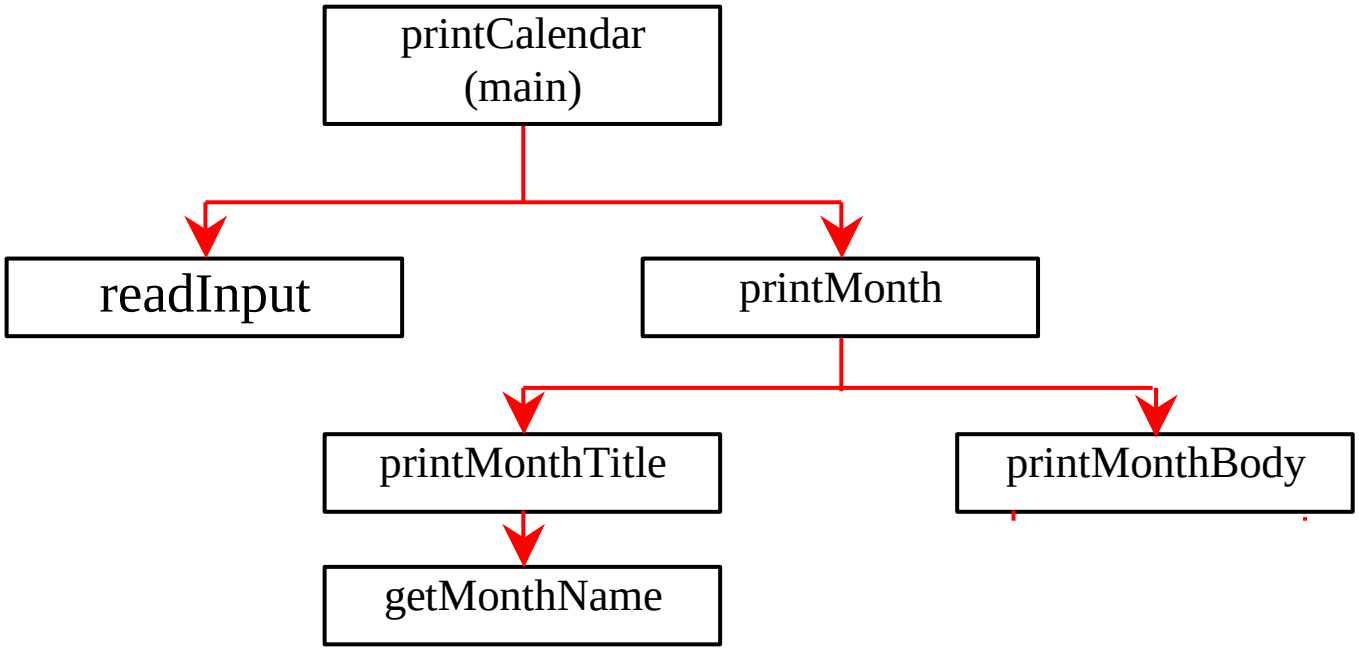


Diagram Tasarımı

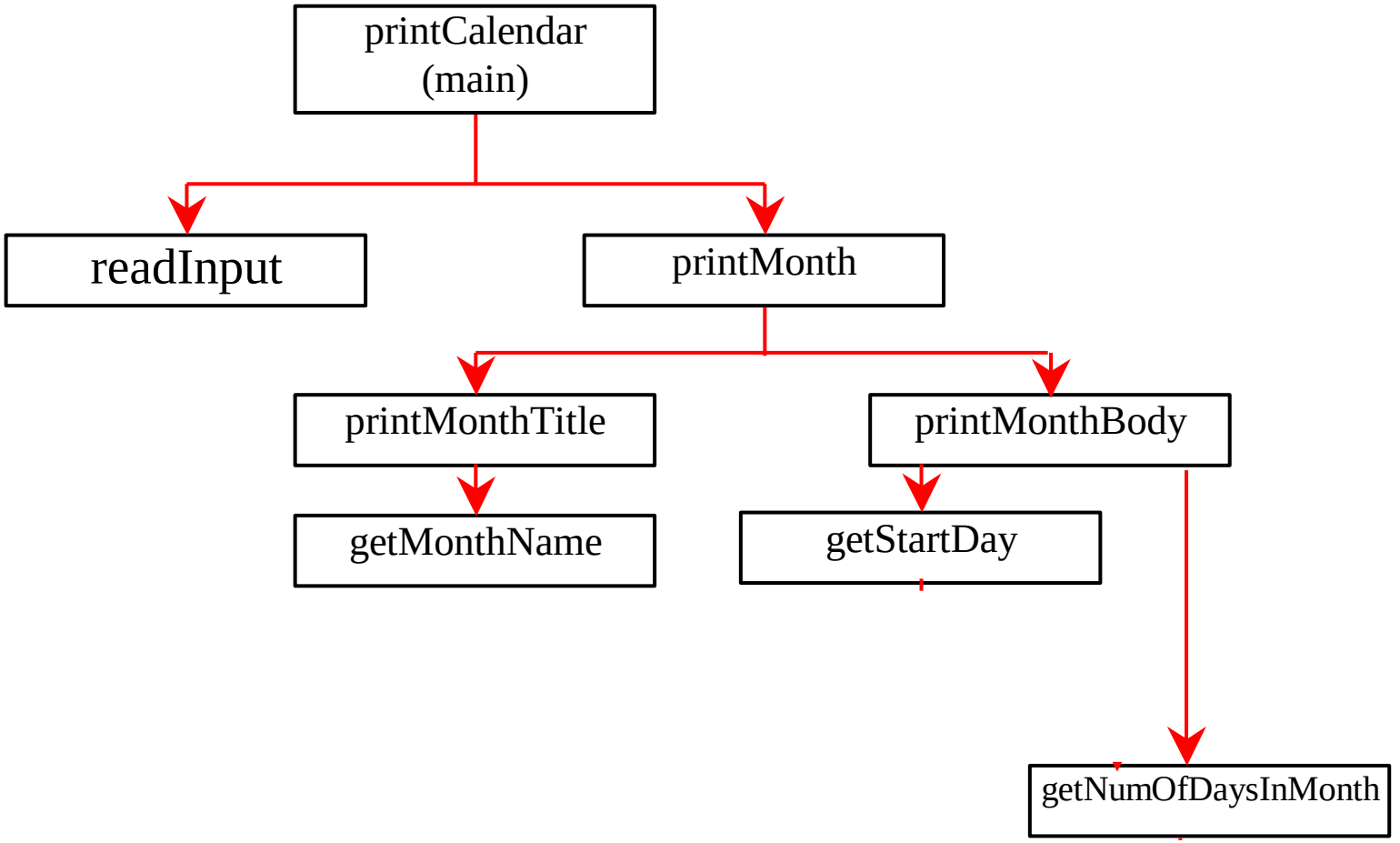


Diagram Tasarımı

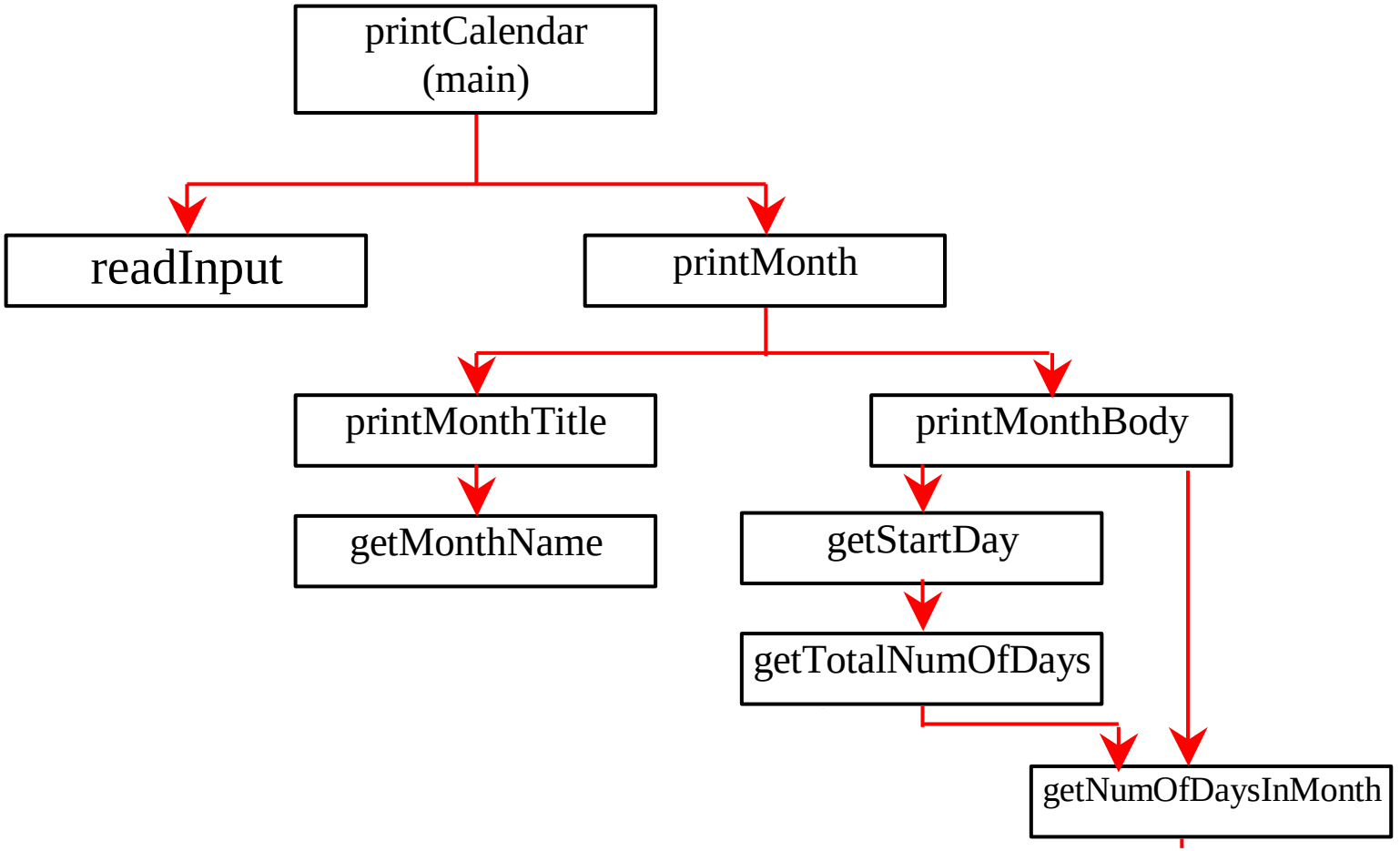
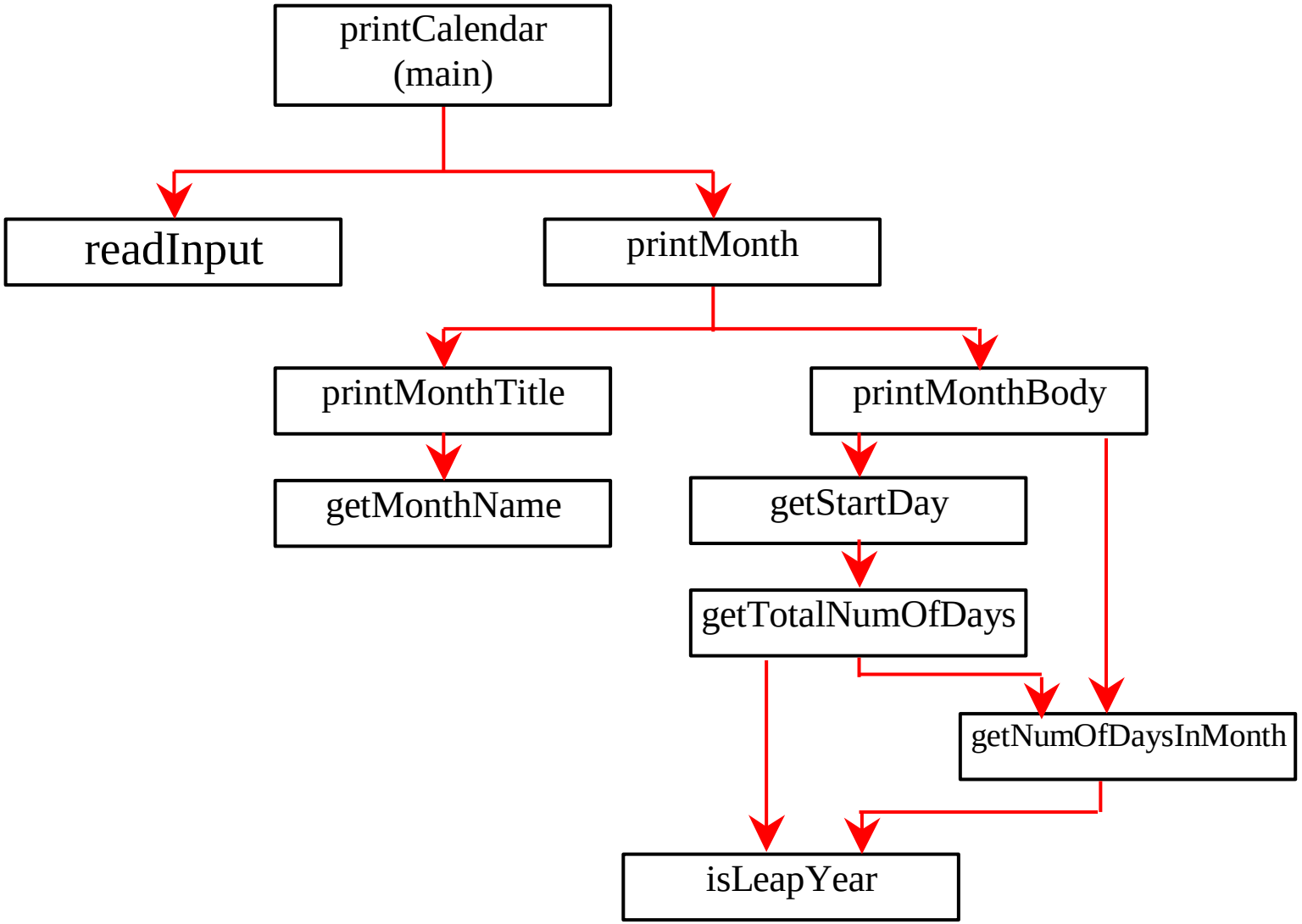


Diagram Tasarımı



Uygulama: Yukarıdan-Aşağı

Yukarıdan aşağıya yaklaşımı, yukarıdan aşağıya doğru diyagramdaki bir metodun uygulanmasıdır. “Stubs” uygulanması için bekleyen metodlar için kullanılır. Yani bir “stub” bir metodun tamamlanmamış basit bir versiyonudur. “stub”ların kullanımı metodun bir arayandan(caller) test edilmesini sağlar. Önce main metodunu uygulayın daha sonra printMonth metodu için stub kullanın. Örneğin, printMonth yıl ve ayı stub içinde göstersin. Böylece programınız şöyle başlayabilir:

A Skeleton for printCalendar

Uygulama: Aşağıdan-Yukarı

Aşağıdan yukarı yaklaşımı, aşağıdan başlayarak yukarı doğru diyagramdaki bir metodun adım adım uygulanmasıdır. Her bir metod uygulandıkça, doğruluğunu test eden program yazınız.

Her iki yaklaşımda iyi çalışır. Metodlarda hata oluşumunu azaltır ve debug etmeyi kolaylaştırır, aşama aşama çalıştırmayı sağlar. Bu iki yaklaşımın birlikte kullanıldığı durumlar da olabilir.