به نام خدا

مبانی کامپیوتر و برنامه سازی

❖ ادامه مبحث متدها

مثال: برنامه ای بنویسید که صد جمله سری فیبوناجی را چاپ کند و اعداد اول آن با * و اعداد کامل آن را با # ، با استفاده از دو متد مشخص کند.

```
public class Fibonacci2 {
   public static void main(String[] args) {
      int a=0,b=1,c;
      System.out.printf("\t%d\t%d",a,b);
      for (int i=1; i<=98;  i++){
         c=a+b;
         System.out.printf("\t%d",c);
         prime(c);
         perfect(c);
         a=b;
         b=c;
      }  //end for i
   }  // end main

   public static void prime(int c) {
         boolean prime=true;
         for (int j=2; j<=c/2 && prime; j++)
                  if (c%j==0)
                     prime=false;
```

```
        if (prime && c!=1)
            System.out.print('*');
    } // end prime

    public static void perfect(int c) {
        int sum=1;
        for (int j=2; j<=c/2; j++)
                if (c%j==0)
                    sum+=j;
        if (sum==c && c!=1)
            System.out.print('#');
    } // end perfect
}  // end Fibonacci2
```

^^

سوال: خروجی برنامه های زیر را مشخص کنید:

```
public class Test {
  public static void main(String[] args) {
    int max = 0;
    max(1, 2, max);
    System.out.println(max);
  }

  public static void max(
      int value1, int value2, int max) {
    if (value1 > value2)
      max = value1;
    else
      max = value2;
  }
}
```

(a)

```
public class Test {
  public static void main(String[] args) {
    int i = 1;
    while (i <= 6) {
      method1(i, 2);
      i++;
    }
  }

  public static void method1(
      int i, int num) {
    for (int j = 1; j <= i; j++) {
      System.out.print(num + " ");
      num *= 2;
    }

    System.out.println();
  }
}
```

(b)

```java
public class Test {
  public static void main(String[] args) {
    // Initialize times
    int times = 3;
    System.out.println("Before the call,"
      + " variable times is " + times);

    // Invoke nPrintln and display times
    nPrintln("Welcome to Java!", times);
    System.out.println("After the call,"
      + " variable times is " + times);
  }

  // Print the message n times
  public static void nPrintln(
      String message, int n) {
    while (n > 0) {
      System.out.println("n = " + n);
      System.out.println(message);
      n--;
    }
  }
}
```

(c)

```java
public class Test {
  public static void main(String[] args) {
    int i = 0;
    while (i <= 4) {
      method1(i);
      i++;
    }

    System.out.println("i is " + i);
  }

  public static void method1(int i) {
    do {
      if (i % 3 != 0)
        System.out.print(i + " ");
      i--;
    }
    while (i >= 1);

    System.out.println();
  }
}
```

(d)

❖ Overloading Methods:

این امکان را فراهم می‌آورند که متدهایی با نام‌های یکسان ولی امضاهای متفاوت داشته باشیم. در فراخوانی این متدهای همنام، با توجه به تعداد و نوع پارامترها فراخوانی متد مربوطه انجام می‌شود.

متد یافتن ماکزیمم چند عدد را در نظر بگیرید، این متد را می توان چندین بار با تعداد و نوع پارامترهای مختلف در یک کلاس تعریف کرد.

TestMethodOverloading.java

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

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

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

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

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

  /** Return the max of three double values */
  public static double max(double num1, double num2, double num3) {
    return max(max(num1, num2), num3);
  }
}
```

چاپ:

```
The maximum of 3 and 4 is 4
The maximum of 3.0 and 5.4 is 5.4
The maximum of 3.0, 5.4, and 10.14 is 10.14
```

تذکر: در مورد فراخوانی max(3,4)، هم متد max (int , int) و هم متد max (double , double) قابل فراخوانی است. کامپایلر بهترین گزینه که max (int , int) است را فراخوانی می کند.

مثال: در مورد زیر کامپایلر خطا می دهد. چرا؟

```
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;
  }
}
```

^^^

سوال: چنانچه متدهای زیر تعریف شده باشند:

```
public static double m(double x, double y)

public static double m(int x, double y)
```

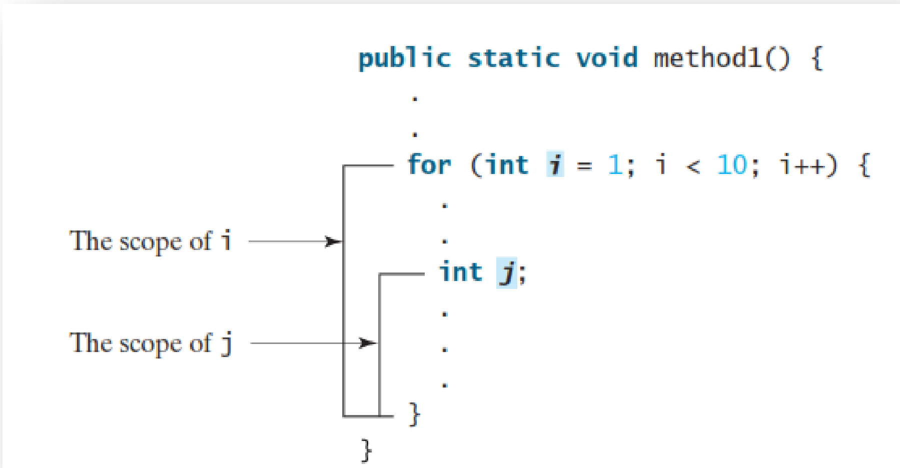
کدام یک از متدهای زیر فراخوانی می شوند:

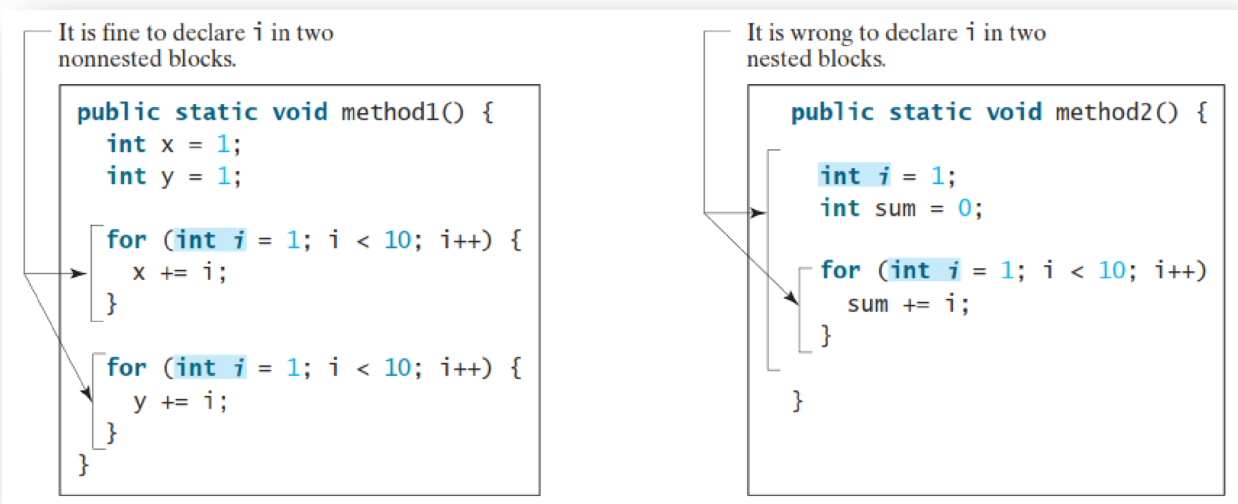
```
a. double z = m(4, 5);

b. double z = m(4, 5.4);

c. double z = m(4.5, 5.4);
```

^^^

❖ محدوده حضور متغیرها The Scope of variables

متغیرها در بلوکی که تعریف می شوند قابل استفاده و دستیابی می باشند.

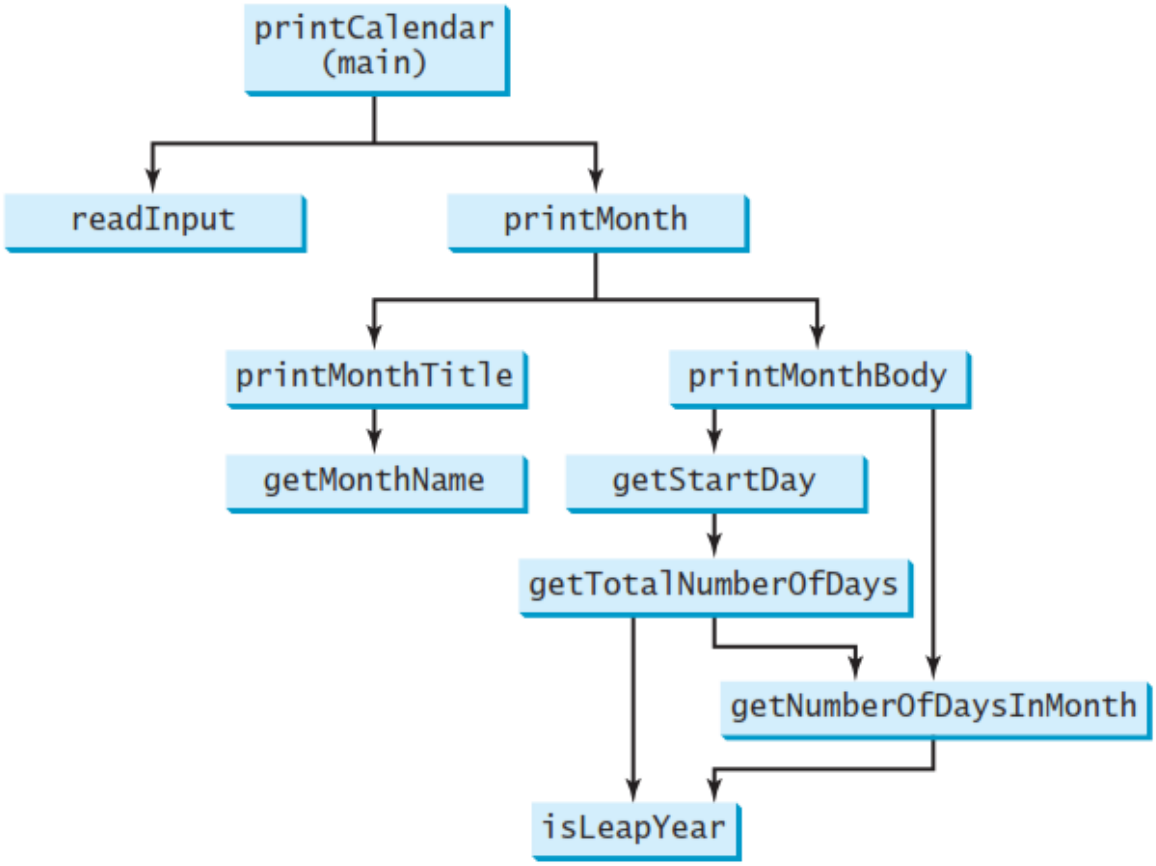




❖ نحوه ارتباط متدها با یکدیگر در یک برنامه

مثال: برنامه چاپ تقویم

سلسه مراتب فراخوانی متدهای برنامه



Benefits of Stepwise Refinement

- **Simpler Program**
- **Reusing Methods**
- **Easier Developing, Debugging, and Testing**
- **Better Facilitating Teamwork**

مزایای پالایش مرحله به مرحله

پالایش مرحله به مرحله یک مسأله بزرگ را به بخش های قابل کنترل کوچک‌تر تقسیم می کند. هر بخش کوچک تر را می توان با استفاده از یک متد پیاده سازی کرد. این رویکرد، نوشتن برنامه، استفاده مجدد، اشکال زدایی، ارزیابی و تست برنامه، و اصلاح برنامه را ساده ترمی کند.

برنامه چاپ تقویم، یک برنامه طولانی است. به جای نوشتن یک دنباله طولانی از کدهای برنامه، روش پالایش مرحله به مرحله، آن را به بخش های کوچک تر تقسیم می کند. این کار، برنامه را ساده تر می کند وخواندن و درک کل برنامه، آسان تر می گردد.

پالایش مرحله به مرحله به استفاده مجدد از کد در یک برنامه کمک می کند، باعث کاهش کدهای اضافی می شود و توسعه برنامه را آسان ترمی کند. از آنجا که هر مسأله فرعی در یک متد حل می شود، می توان متدها را توسعه داد و به طور جداگانه اشکال زدایی و آزمایش کرد. این کارخطاها را جدا می کند و توسعه، اشکال زدایی و آزمایش را آسان تر می کند، همچنین باعث صرفه جویی در وقت می شود.

با استفاده از پالایش مرحله به مرحله، کار گروهی روی یک برنامه ساده تر می گردد. هر بخش، می تواند به یک برنامه نویس داده شود و برنامه نویس کد مربوطه را تولید کند.

PrintCalendar.java

```java
import java.util.Scanner;

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

    // Prompt the user to enter year
    System.out.print("Enter full year (e.g., 2012): ");
    int year = input.nextInt();

    // Prompt the user to enter month
    System.out.print("Enter month as a number between 1 and 12: ");
    int month = input.nextInt();

    // Print calendar for the month of the year
    printMonth(year, month);
  }

  /** Print the calendar for a month in a year */
  public static void printMonth(int year, int month) {
    // Print the headings of the calendar
    printMonthTitle(year, month);

    // Print the body of the calendar
    printMonthBody(year, month);
  }

  /** Print the month title, e.g., March 2012 */
  public static void printMonthTitle(int year, int month) {
    System.out.println("         " + getMonthName(month)
      + " " + year);
    System.out.println("-----------------------------");
    System.out.println(" Sun Mon Tue Wed Thu Fri Sat");
  }

  /** Get the English name for the month */
  public static String getMonthName(int month) {
    String monthName = "";
    switch (month) {
      case 1: monthName = "January"; break;
      case 2: monthName = "February"; break;
      case 3: monthName = "March"; break;
      case 4: monthName = "April"; break;
      case 5: monthName = "May"; break;
      case 6: monthName = "June"; break;
      case 7: monthName = "July"; break;
      case 8: monthName = "August"; break;
      case 9: monthName = "September"; break;
      case 10: monthName = "October"; break;
      case 11: monthName = "November"; break;
      case 12: monthName = "December";
    }

    return monthName;
  }

  /** Print month body */
```

```
59    public static void printMonthBody(int year, int month) {
60      // Get start day of the week for the first date in the month
61      int startDay = getStartDay(year, month)
62
63      // Get number of days in the month
64      int numberOfDaysInMonth = getNumberOfDaysInMonth(year, month);
65
66      // Pad space before the first day of the month
67      int i = 0;
68      for (i = 0; i < startDay; i++)
69        System.out.print("    ");
70
71      for (i = 1; i <= numberOfDaysInMonth; i++) {
72        System.out.printf("%4d", i);
73
74        if ((i + startDay) % 7 == 0)
75          System.out.println();
76      }
77
78      System.out.println();
79    }
80
81    /** Get the start day of month/1/year */
82    public static int getStartDay(int year, int month) {
83      final int START_DAY_FOR_JAN_1_1800 = 3;
84      // Get total number of days from 1/1/1800 to month/1/year
85      int totalNumberOfDays = getTotalNumberOfDays(year, month);
86
87      // Return the start day for month/1/year
88      return (totalNumberOfDays + START_DAY_FOR_JAN_1_1800) % 7;
89    }
90
91    /** Get the total number of days since January 1, 1800 */
92    public static int getTotalNumberOfDays(int year, int month) {
93      int total = 0;
94
95      // Get the total days from 1800 to 1/1/year
96      for (int i = 1800; i < year; i++)
97        if (isLeapYear(i))
98          total = total + 366;
99        else
100         total = total + 365;
101
102     // Add days from Jan to the month prior to the calendar month
103     for (int i = 1; i < month; i++)
104       total = total + getNumberOfDaysInMonth(year, i);
105
106     return total;
107   }
108
109   /** Get the number of days in a month */
110   public static int getNumberOfDaysInMonth(int year, int month) {
111     if (month == 1 || month == 3 || month == 5 || month == 7 ||
112       month == 8 || month == 10 || month == 12)
113       return 31;
114
115     if (month == 4 || month == 6 || month == 9 || month == 11)
116       return 30;
117
118     if (month == 2) return isLeapYear(year) ? 29 : 28;
```

```java

    return 0; // If month is incorrect
  }

  /** Determine if it is a leap year */
  public static boolean isLeapYear(int year) {
    return year % 400 == 0 || (year % 4 == 0 && year % 100 != 0);
  }
}
```

^^^