متدهای جاوا

متد جاوا مجموعه ای از وضعیت هاست که برای اجرای یک عملکرد گروه بندی می شوند. به عنوان مثال :وقتی متود System.out.println را فرا خوانی می کنید، درواقع سیستم چندین عبارت را برای نمایش یک پیغام در console اجرا می کند.

اکنون شما چگونگی ایجاد متودهای خود را با بازگشت مقادیر یا بدون بازگشت مقادیر، فراخوانی یک متود با پارامترها و بدون پارامترها، بارگذاری متودها با استفاده از همان نام ها و به کارگیری متود انتزاع در طراحی برنامه فرا خواهید گرفت.

ایجاد متد:

با توجه به مثال :زیر می توانید ترکیب یک متود را توضیح دهید:

مثال :

```
public static int funcName(int a, int b)

{

  // body

}
```

در اینجا:

public static: اصلاح کننده · int: نوع بازگشت · funcName: نام عملکرد · a, b: پارامترهای فرمال

int a, int b: لیست پارامترها

متودها با عنوان Procedures یا Functions نیز شناخته می شوند:

Procedures: هیچ مقداری را بازنمی گردانند.

Functions: یک مقدار را بازمی گردانند.

تعریف متود حاوی یک تیتر و یک بدنه می شود. همین مورد در زیر نشان داده شده است:

مثال :

```
modifier returnType nameOfMethod (Parameter List) {

 // method body
```

```
}
```

ترکیب نمایش داده شده در بالا حاوی:

modifier: ·نوع access مربوط به متود را تعریف می کند و استفاده از آن انتخابی می باشد:returnType · .متود ممکن است یک مقدار را بازگرداند :nameOfMethod · .این نام متود می باشد. ویژگی خاص متود حاوی نام متود و لیست پارامترها می باشد :Parameter List · .لیست پارامترها که حاوی نوع، ترتیب و تعداد پارامترهای یک متود می باشد. اینها انتخابی هستند، متود ممکن حاوی هیچ پارامتری نباشد :method body · .بدنه ی متود تعریف می کند که متود با عبارات چه انجام می دهد.

مثال :

در اینجا source code متود تعریف شده ی بالا به نام ()max می باشد. این متود دارای دو پارامتر num1 و num2 می باشد و حداکثر را بین هر دو بازمی گرداند:

```
/** the snippet returns the minimum between two numbers */

public static int minFunction(int n1, int n2) {

   int min;

   if (n1 > n2)

      min = n2;

   else

      min = n1;


   return min;

}
```

فراخوانی متد:

برای استفاده از یک متود ابتدا باید فراخوانده شود. دو روش در فراخوانی یک متود وجود دارد، به عنوان مثال :متودی که یک مقدار را بازمی گرداند و متودی که هیچ چیز بازنمی گرداند.

فرایند فراخوانی ماود ساده می باشد. وقتی که برنامه ای یک متود را فرامی خواند، کنترل برنامه به فراخوانی متود تغییر می یابد. سپس این متود فراخوانده شده کنترل را به فراخواننده تحت دو شرط، بازمی گرداند، وقتی:

·بازگشت عبارت اجرا می شود. · اتمام متود در حال بستن گروه می باشد .متودهایی که پوچ (void) گزارش می دهند، برای فراخوانی یک عبارت در نظر گرفته می شوند. اجازه بدهید مثالی را در نظر بگیریم:

```
System.out.println("This is tutorialspoint.com!");
```

مقدار متد returning با مثال زیر قابل درک می باشد:

```
int result = sum(6, 9);
```

در زیر مثالی از چگونگی تعریف یک متود و چگونگی فراخوانی آن را مشاهده می کنید:

مثال :

```
public class ExampleMinNumber{

   public static void main(String[] args) {
      int a = 11;
      int b = 6;
      int c = minFunction(a, b);
      System.out.println("Minimum Value = " + c);
   }

   /** returns the minimum of two numbers */
   public static int minFunction(int n1, int n2) {
      int min;
      if (n1 > n2)
         min = n2;
      else
         min = n1;

      return min;
   }
```

```
}
```

این مثال :نتیجه ی زیر را تولید خواهد کرد:

Minimum value = 6

لغت کلیدی:void

لغت کلیدی void به ما اجازه می دهد تا متودهایی را ایجاد کنیم که هیچ مقداری را بازنمی گرداند. در مثال :زیر یک متود methodRankPointsرا در نظر می گیریم. این متود یک متود void می باشد که هیچ مقداری را بازنمی گرداند. فراخوانی به یک متود void باید یک عبارت باشد، به عنوان مثال(methodRankPoints(255.7 :، که یک عبارت جاواست که با یک نقطه ویرگول به پایان می رسد، همانطور که در مثال :زیر نشان داده شده است:

مثال :

```
public class ExampleVoid {


   public static void main(String[] args) {

      methodRankPoints(255.7);

   }


   public static void methodRankPoints(double points) {

      if (points >= 202.5) {

         System.out.println("Rank:A1");

      }

      else if (points >= 122.4) {

         System.out.println("Rank:A2");

      }

      else {

         System.out.println("Rank:A3");

      }

   }
```

```
}
```

این مثال :نتیجه ی زیر را تولید خواهد کرد:

Rank:A1

انتقال پارامترها با مقدار:

هنگام کار تحت فرایند فراخوانی، argumentباید منتقل شود. اینها باید به همان ترتیبی باشند که پارامترهای مربوطه در تعیین متود هستند. پارامترها می توانند با مقدار یا با مرجع منتقل شوند.

انتقال پارامترها با مقدار به معنای فراخوانی یک متود با یک پارامتر می باشد. از این طریق مقدار argument به پارامتر منتقل می شود.

برنامه ی زیر مثالی از انتقال پارامتر با مقدار را نشان می دهد. مقادیر argumentها، حتی پس از فراخوانی متود، بدون تغییر باقی می مانند.

```
public class swappingExample {


   public static void main(String[] args) {

      int a = 30;

      int b = 45;


      System.out.println("Before swapping, a = " +

                         a + " and b = " + b);


      // Invoke the swap method

      swapFunction(a, b);

      System.out.println("\n**Now, Before and After swapping values will be
same here**:");

      System.out.println("After swapping, a = " +

                         a + " and b is " + b);

   }
```

```
    public static void swapFunction(int a, int b) {

        System.out.println("Before swapping(Inside), a = " + a
                           + " b = " + b);
        // Swap n1 with n2
        int c = a;
        a = b;
        b = c;

        System.out.println("After swapping(Inside), a = " + a
                           + " b = " + b);
    }
}
```

این مثال :نتیجه ی زیر را به دنبال دارد:

```
Before swapping, a = 30 and b = 45

Before swapping(Inside), a = 30 b = 45

After swapping(Inside), a = 45 b = 30


**Now, Before and After swapping values will be same here**:

After swapping, a = 30 and b is 45
```

Metho Overloading:

وقتی که یک گروه دارای دو و یا چند متود هم نام اما با پارامترهای متفاوت می باشد، این امر method overloading نامیده می شود، که متفاوت با overridingمی باشد. در overriding یک متود دارای همان نام، نوع، تعداد پارامترها و غیره می باشد.

اجازه بدهید مثال :قبل را برای یافتن حداقل تعداد عدد صحیح در نظر بگیریم. اجازه بدهید بگوییم که می خواهیم حداقل تعداد نوع double را پیدا کنیم. سپس مفهوم overloading برای ایجاد دو یا چند متود هم نام اما با پارامترهای متفاوت، معرفی خواهد شد.

مثال :زیر همین مورد را توضیح می دهد:

```
public class ExampleOverloading{


   public static void main(String[] args) {
      int a = 11;
      int b = 6;
      double c = 7.3;
      double d = 9.4;
      int result1 = minFunction(a, b);
      // same function name with different parameters
      double result2 = minFunction(c, d);
      System.out.println("Minimum Value = " + result1);
      System.out.println("Minimum Value = " + result2);
   }


  // for integer
   public static int minFunction(int n1, int n2) {
      int min;
      if (n1 > n2)
         min = n2;
      else
         min = n1;


      return min;
```

```
    }
    // for double
    public static double minFunction(double n1, double n2) {
      double min;
       if (n1 > n2)
          min = n2;
       else
          min = n1;

       return min;
    }
}
```

متد جاوا مجموعه ای از وضعیت هاست که برای اجرای یک عملکرد گروه بندی می شوند. به عنوان مثال :وقتی متود System.out.println را فرا خوانی می کنید، درواقع سیستم چندین عبارت را برای نمایش یک پیغام در console اجرا می کند.

اکنون شما چگونگی ایجاد متودهای خود را با بازگشت مقادیر یا بدون بازگشت مقادیر، فراخوانی یک متود با پارامترها و بدون پارامترها، بارگذاری متودها با استفاده از همان نام ها و به کارگیری متود انتزاع در طراحی برنامه فرا خواهید گرفت.

ایجاد متد:

با توجه به مثال :زیر می توانید ترکیب یک متود را توضیح دهید:

مثال :

```
public static int funcName(int a, int b)
{
  // body
}
```

در اینجا:

public static: ·اصلاح کننده int: · نوع بازگشت funcName: · نام عملکرد a, b: · پارامترهای فرمال int a, int b: لیست پارامترها

متودها با عنوان Procedures یا Functions نیز شناخته می شوند:

Procedures : ·هیچ مقداری را بازنمی گرداند.

Functions: ·یک مقدار را بازمی گردانند.

تعریف متود حاوی یک تیتر و یک بدنه می شود. همین مورد در زیر نشان داده شده است:

مثال :

```
modifier returnType nameOfMethod (Parameter List) {

 // method body

}
```

ترکیب نمایش داده شده در بالا حاوی:

modifier: ·نوع access مربوط به متود را تعریف می کند و استفاده از آن انتخابی می باشد:returnType · .متود ممکن است یک مقدار را بازگرداند nameOfMethod: · .این نام متود می باشد. ویژگی خاص متود حاوی نام متود و لیست پارامترها می باشد Parameter List: · .لیست پارامترها که حاوی نوع، ترتیب و تعداد پارامترهای یک متود می باشد. اینها انتخابی هستند، متود ممکن حاوی هیچ پارامتری نباشد :method body · .بدنه ی متود تعریف می کند که متود با عبارات چه انجام می دهد.

مثال :

در اینجا source code متود تعریف شده ی بالا به نام max() می باشد. این متود دارای دو پارامتر num1 و num2 می باشد و حداکثر را بین هر دو بازمی گرداند:

```
/** the snippet returns the minimum between two numbers */

public static int minFunction(int n1, int n2) {

   int min;

   if (n1 > n2)

      min = n2;
```

```
   else
      min = n1;

   return min;
}
```

فراخوانی متد:

برای استفاده از یک متود ابتدا باید فراخوانده شود. دو روش در فراخوانی یک متود وجود دارد، به عنوان مثال :متودی که یک مقدار را بازمی گرداند و متودی که هیچ چیز بازنمی گرداند.

فرایند فراخوانی ماود ساده می باشد. وقتی که برنامه ای یک متود را فرامی خواند، کنترل برنامه به فراخوانی متود تغییر می یابد. سپس این متود فراخوانده شده کنترل را به فراخواننده تحت دو شرط، بازمی گرداند، وقتی:

·بازگشت عبارت اجرا می شود. · اتمام متود در حال بستن گروه می باشد .متودهایی که پوچ (void) گزارش می دهند، برای فراخوانی یک عبارت در نظر گرفته می شوند. اجازه بدهید مثالی را در نظر بگیریم:

```
System.out.println("This is tutorialspoint.com!");
```

مقدار متد returning با مثال :زیر قابل درک می باشد:

```
int result = sum(6, 9);
```

در زیر مثالی از چگونگی تعریف یک متود و چگونگی فراخوانی آن را مشاهده می کنید:

مثال :

```
public class ExampleMinNumber{

   public static void main(String[] args) {
      int a = 11;
      int b = 6;
      int c = minFunction(a, b);
      System.out.println("Minimum Value = " + c);
   }
```

```
    /** returns the minimum of two numbers */
    public static int minFunction(int n1, int n2) {
        int min;
        if (n1 > n2)
            min = n2;
        else
            min = n1;

        return min;
    }
}
```

این مثال :نتیجه ی زیر را تولید خواهد کرد:

Minimum value = 6

لغت کلیدی:void

لغت کلیدی void به ما اجازه می دهد تا متودهایی را ایجاد کنیم که هیچ مقداری را بازنمی گرداند. در مثال :زیر یک متود methodRankPointsرا در نظر می گیریم. این متود یک متود void می باشد که هیچ مقداری را بازنمی گرداند.
فراخوانی به یک متود void باید یک عبارت باشد، به عنوان مثال methodRankPoints(255.7) :، که یک عبارت جاواست که با یک نقطه ویرگول به پایان می رسد، همانطور که در مثال :زیر نشان داده شده است:

مثال :

```
public class ExampleVoid {

    public static void main(String[] args) {
        methodRankPoints(255.7);
    }
```

```
    public static void methodRankPoints(double points) {
        if (points >= 202.5) {
            System.out.println("Rank:A1");
        }
        else if (points >= 122.4) {
            System.out.println("Rank:A2");
        }
        else {
            System.out.println("Rank:A3");
        }
    }
}
```

این مثال :نتیجه ی زیر را تولید خواهد کرد:

Rank:A1

انتقال پارامترها با مقدار:

هنگام کار تحت فرایند فراخوانی، argumentباید منتقل شود. اینها باید به همان ترتیبی باشند که پارامترهای مربوطه در تعیین متود هستند. پارامترها می توانند با مقدار یا با مرجع منتقل شوند.

انتقال پارامترها با مقدار به معنای فراخوانی یک متود با یک پارامتر می باشد. از این طریق مقدار argument به پارامتر منتقل می شود.

برنامه ی زیر مثالی از انتقال پارامتر با مقدار را نشان می دهد. مقادیر argumentها، حتی پس از فراخوانی متود، بدون تغییر باقی می مانند.

```
public class swappingExample {

    public static void main(String[] args) {
        int a = 30;
        int b = 45;
```

```
        System.out.println("Before swapping, a = " +
                           a + " and b = " + b);

        // Invoke the swap method
        swapFunction(a, b);
        System.out.println("\n**Now, Before and After swapping values will be
same here**:");
        System.out.println("After swapping, a = " +
                           a + " and b is " + b);
    }

    public static void swapFunction(int a, int b) {

        System.out.println("Before swapping(Inside), a = " + a
                           + " b = " + b);
        // Swap n1 with n2
        int c = a;
        a = b;
        b = c;

        System.out.println("After swapping(Inside), a = " + a
                           + " b = " + b);
    }
}
```

این مثال نتیجه ی زیر را به دنبال دارد:

```
Before swapping, a = 30 and b = 45

Before swapping(Inside), a = 30 b = 45

After swapping(Inside), a = 45 b = 30


**Now, Before and After swapping values will be same here**:

After swapping, a = 30 and b is 45
```

Metho Overloading:

وقتی که یک گروه دارای دو و یا چند متود هم نام اما با پارامترهای متفاوت می باشد، این امر method overloading نامیده می شود، که متفاوت با overridingمی باشد. در overriding یک متود دارای همان نام، نوع، تعداد پارامترها و غیره می باشد.

اجازه بدهید مثال :قبل را برای یافتن حداقل تعداد عدد صحیح در نظر بگیریم. اجاره بدهید بگوییم که می خواهیم حداقل تعداد نوع double را پیدا کنیم. سپس مفهوم overloading برای ایجاد دو یا چند متود هم نام اما با پارامترهای متفاوت، معرفی خواهد شد.

مثا :زیر همین مورد را توضیح می دهد:

```
public class ExampleOverloading{


   public static void main(String[] args) {

      int a = 11;

      int b = 6;

      double c = 7.3;

      double d = 9.4;

      int result1 = minFunction(a, b);

      // same function name with different parameters

      double result2 = minFunction(c, d);

      System.out.println("Minimum Value = " + result1);

      System.out.println("Minimum Value = " + result2);
```

```
    }


   // for integer
    public static int minFunction(int n1, int n2) {
       int min;
       if (n1 > n2)
          min = n2;
       else
          min = n1;


       return min;
    }
    // for double
    public static double minFunction(double n1, double n2) {
      double min;
       if (n1 > n2)
          min = n2;
       else
          min = n1;


       return min;
    }
}
```

این<img src="/Administrator/Image/code.png" alt=""> مثال: نتیجه ی زیر را به دنبال دارد:

```
Minimum Value = 6

Minimum Value = 7.3
```

متدهای overloading برنامه را خوانا می سازد. در اینجا دارای یک نام می باشند، اما پارامترهای متفاوت دارند. حداقل تعداد انواع صحیح(integer) و doubleنتیجه می باشد.

استفاده از argument های Command-Line:

گاهی اوقات وقتی برنامه ای را اجرا می کنید، تمایل خواهید داشت که اطلاعات را به برنامه انتقال دهید. این کار به وسیله ی انتقال خط فرمان argumentها به main( ) انجام می شود.

یک command-line argument اطلاعاتی است که مستقیما نام برنامه را در هنگام اجرا روی خط فرمان دنبال می کند. دسترسی به argument های خط فرمان در داخل یک برنامه ی جاوا بسیار آسان می باشد، آنها مانند رشته ها در ردیف Stringبه main( ) منتقل می شوند.

مثال:

برنامه ی زیر همه ی argument های خط فرمان را که از طریق آنها فراخوانده شد، نمایش می دهد:

```
public class CommandLine {


   public static void main(String args[]){

      for(int i=0; i
```

اجرای این برنامه را به شکلی که در زیر نشان داده شده، امتحان کنید:

```
java CommandLine this is a command line 200 -100
```

این برنامه نتیجه ی زیر را به دنبال دارد:

```
args[0]: this

args[1]: is

args[2]: a

args[3]: command

args[4]: line

args[5]: 200

args[6]: -100
```

سازنده ها:

یک سازنده در هنگام ایجاد، یک آبجکت را مقدار دهی می کند. این سازنده هم نام گروه خود می باشد و از لحاظ نحوی مشابه یک متود است. به هرحال سازنده ها هیچ نوع بازگشتی مشخصی ندارند.

معمولا برای دادن مقادیر اولیه به متغیرهای نمونه که توسط گروه تعریف شده اند ، یا برای اجرای دیگر روش های راه اندازی که برای ایجاد یک آبجکت کامل ، از یک سازنده استفاده خواهید کرد.

همه ی گروه ها دارای سازنده هستند، چه شما برای آن تعریف کنید و چه نکنید، زیرا جاوا به طور خودکار یک سازنده ی پیش فرض ارائه می دهد که همه ی متغیرهای عضو را به صفر مقدار دهی می کند. به هرحال زمانی که شما سازنده ی خود را تعریف می کنید، سازنده ی پیش فرض دیگر مورد استفاده قرار نمی گیرد.

در اینجا مثال ساده ای را می بینید که از یک سازنده استفاده می کند.

```
// A simple constructor.

class MyClass {

   int x;
   // Following is the constructor

   MyClass() {

      x = 10;

   }

}
```

سازنده را برای آبجکت های اولیه فرا می خوانید، مانند زیر:

```
public class ConsDemo {
    public static void main(String args[]) {

      MyClass t1 = new MyClass();

      MyClass t2 = new MyClass();

      System.out.println(t1.x + " " + t2.x);

   }

}
```

اکثر اوقات، به سازنده ای نیاز خواهید داشت که یک یا چند پارامتر را می پذیرد. پارامترها به همان روشی به یک سازنده اضافه می شوند که به یک متود اضافه می شوند، تنها کافیست آنها را در داخل پرانتزها و پس از نام سازنده قرار دهید.

در اینجا مثال ساده ای را مشاهده می کنید که از یک سازنده استفاده می کند:

```
// A simple constructor.

class MyClass {

   int x;
   // Following is the constructor

   MyClass(int i ) {

      x = i;

   }

}
```

سازنده را برای مقدار دهی آبجکت ها فرا می خوانید، مانند زیر:

```
public class ConsDemo {
   public static void main(String args[]) {

      MyClass t1 = new MyClass( 10 );

      MyClass t2 = new MyClass( 20 );

      System.out.println(t1.x + " " + t2.x);

   }

}
```

این مثال نتیجه ی زیر را تولید می کند:

```
Variable Arguments(var-args)
```

JDK 1.5شما را قادر به انتقال یک متغیر عددی از argument های هم نوع به یک متود می سازد. پارامتر در متود مانند زیر اعلام می شود:

typeName... parameterName

در اعلام متود، نوع را که با یک (...) دنبال می شود، تعیین می کنید. فقط یک پارامتر variable-length ممکن است در یک متود تعیین شود و این پارامتر باید آخرین پارامتر باشد. هر پارامتر معمول دیگری باید قبل از آن قرار بگیرد.

```
public class VarargsDemo {
   public static void main(String args[]) {

      // Call method with variable args
```

```
        printMax(34, 3, 3, 2, 56.5);
    printMax(new double[]{1, 2, 3});
  }

  public static void printMax( double... numbers) {
  if (numbers.length == 0) {
    System.out.println("No argument passed");
    return;
  }

  double result = numbers[0];

  for (int i = 1; i <  numbers.length; i++)
    if (numbers[i] >  result)
    result = numbers[i];
    System.out.println("The max value is " + result);
  }
}
```

این مثال :نتیجه ی زیر را به دنبال دارد: The max value is 56.5 The max value is 3.0

متود:( )finalize

ممکن است متودی را تعریف کنید که درست قبل از تخریب نهایی یک آبجکت به وسیله ی garbage collector ، فراخوانده خواهد شد. این متود ( )finalize نامیده می شود و می تواند برای اطمینان دادن این مسئله استفاده شود که آیا به درستی به پایان رسیده.

برای مثال :ممکن است از ( )finalize برای اطمینان از بسته شدن یک فایل باز در آن آبجکت استفاده کنید. هروقت لازم است آبجکتی از آن گروه را بازیابی کنند، زمان اجرای جاوا آن برنامه را فرا می خواند.

در داخل متود finalize( ) ، فعالیت هایی را تعیین می کنید که باید قبل از تخریب یک آبجکت اجرا شوند.

متود finalize( ) دارای این فرم کلی می باشد:

```
protected void finalize( )
{
   // finalization code here
}
```

در اینجا لغت کلیدی محافظت شده یک تعیین کننده است که با کد تعریف شده در خارج گروه آن، مانع دسترسی به finalize( )می شود.

این به این معناست که شما از زمان اجرای finalize( ) و حتی از اجرای آن مطلع نمی شوید. برای مثال :اگر برنامه ی شما قبل از وقوع garbage collection اتفاق بیفتد، finalize( )اجرا نخواهد شد.

این مثال :نتیجه ی زیر را به دنبال دارد Minimum Value = 6 Minimum Value = 7.3 :

متدهای overloading برنامه را خوانا می سازد. در اینجا دارای یک نام می باشند، اما پارامترهای متفاوت دارند. حداقل تعداد انواع صحیح(integer) و doubleنتیجه می باشد.

استفاده از argument های:Command-Line

گاهی اوقات وقتی برنامه ای را اجرا می کنید، تمایل خواهید داشت که اطلاعات را به برنامه انتقال دهید. این کار به وسیله ی انتقال خط فرمان argumentها به main( ) انجام می شود.

یک command-line argument اطلاعاتی است که مستقیما نام برنامه را در هنگام اجرا روی خط فرمان دنبال می کند. دسترسی به argument های خط فرمان در داخل یک برنامه ی جاوا بسیار آسان می باشد، آنها مانند رشته ها در ردیف Stringبه main( ) منتقل می شوند.

مثال: :

برنامه ی زیر همه ی argument های خط فرمان را که از طریق آنها فراخوانده شد، نمایش می دهد:

```
public class CommandLine {
   public static void main(String args[]){

      for(int i=0; i
```

اجرای این برنامه را به شکلی که در زیر نشان داده شده، امتحان کنید:

java CommandLine this is a command line 200 -100

این برنامه نتیجه ی زیر را به دنبال دارد:

```
args[0]: this
args[1]: is
args[2]: a
args[3]: command
args[4]: line
args[5]: 200
args[6]: -100
```

TAHLILGA INSTITUTE
آموزشگاه تحلیلگر داده ها

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد7

88146323 - 88446780 - 88146330

http://www.tahlildadeh.com/