İleri Programlama

Fonksiyonlar

Hüseyin Ahmetoğlu

Neden Fonksiyonlar?

- Zaman zaman, kodun belirli bir kısmı birçok kez kullanılmalıdır. Kodu birçok kez yeniden yazmak yerine, onları bir "alt yordama" koymak ve bu "alt yordamı" defalarca "çağırmak" - bakım ve anlama kolaylığı açısından daha iyidir.
- Yordam, Yöntem, Metot, Fonksiyon, Alt program

Metot kullanmanın faydaları

- Böl ve fethet: Programı basit, küçük parçalardan veya bileşenlerden oluşturun. Programı bağımsız görevler halinde modüler hale getirin.
- Kodu tekrarlamaktan kaçının: Kopyalamak ve yapıştırmak kolaydır, ancak tüm kopyaların bakımı ve senkronize edilmesi zordur.
- Yazılımın Yeniden Kullanımı: Diğer programlardaki metotları kütüphanelere (veya API) paketleyerek yeniden kullanabilirsiniz.

Metot Kullanma

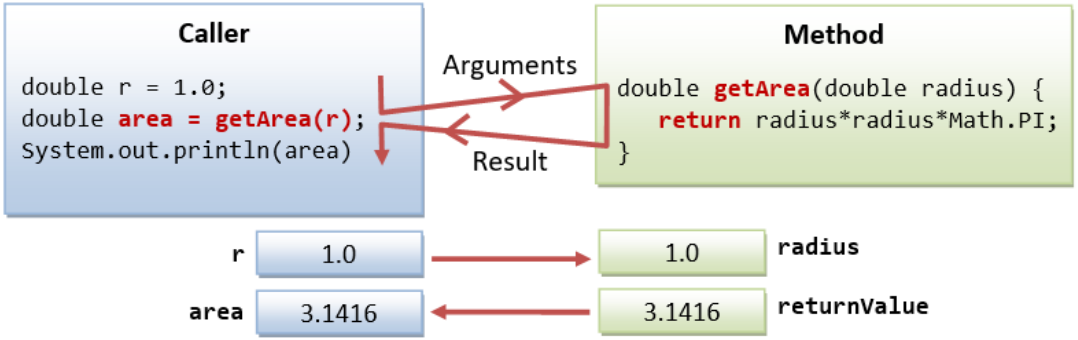
- Çağırıcı, bir metodu çağırır ve metoda argümanlar iletir.
- Metot:
  - Çağırıcı tarafından iletilen argümanları alır, metodun gövdesinde tanımlanan programlanmış işlemleri gerçekleştirir ve Çağırıcıya bir sonuç döndürür.
- Çağırıcı sonucu alır ve işlemlerine devam eder.

Metot Kullanma

```
area is 3.8013271108436504
area 2 is 15.205308443374602
area 3 is 34.21194399759284
```

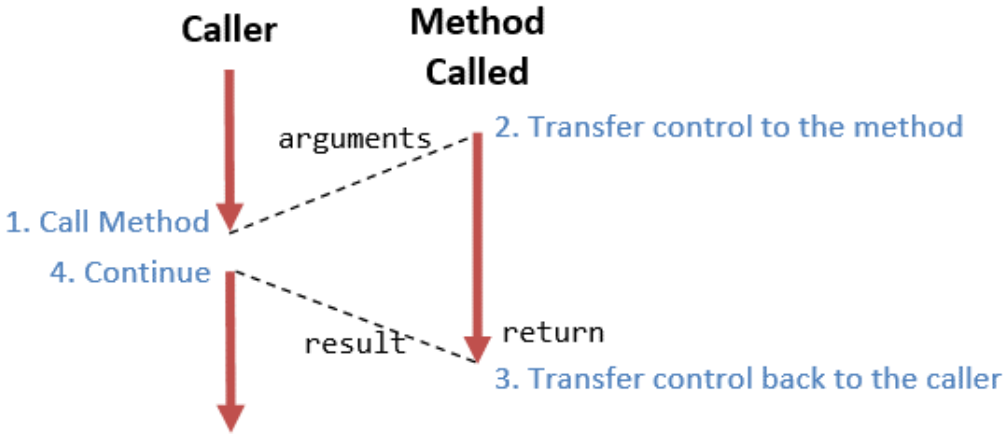
```java
public class EgMethodGetArea {
   // The entry main method
   public static void main(String[] args) {
      double r = 1.1, area, area2;
      // Call (Invoke) method getArea() and return
      area = getArea(r);
      System.out.println("area is " + area);
      // Call method getArea() again and return
      area2 = getArea(2.2);
      System.out.println("area 2 is " + area2);
      // Call method getArea() one more time and return
      System.out.println("area 3 is " + getArea(3.3));
   }

   // Method getArea() Definition.
   // Compute and return the area (in double) of circle given its radius (in double).
   public static double getArea(double radius) {
      return radius * radius * Math.PI;
   }
}
```

Metot Kullanma




```
area is 3.8013271108436504
area 2 is 15.205308443374602
area 3 is 34.21194399759284
```

Metot Kullanma



Metot Tanımlama Syntax

```
public static returnValueType methodName(arg-1-type arg-1, arg-2-type arg-2,... ) {
   body ;
}
```

```
// Examples
// Return circle's area given its radius
public static double getArea(double radius) {
   return radius * radius * Math.PI;
}

// Return maximum among two given integers
public static int max(int number1, int number2) {
   if (number1 > number2) {
      return number1;
   } else {
      return number2;
   }
}
```

Metot Çağırma

```
// Calling getArea()
double area1 = getArea(1.1);  // with literal as argument
double r2 = 2.2;
double area2 = getArea(r2);   // with variable as argument
double r3 = 3.3;
System.out.println("Area is: " + area(r3));

// Calling max()
int result1 = max(5, 8);
int i1 = 7, i2 = 9;
int result2 = max(i1, i2);
System.out.println("Max is: " + max(15, 16));
```

Metot Çağırma

```
/** Example of Java Method definition and invocation */
public class EgMinMaxMethod {
   // The entry main() method
   public static void main(String[] args) {
      int a = 6, b = 9, max, min;
      max = max(a, b);  // invoke method max() with arguments
      min = min(a, b);  // invoke method min() with arguments
      System.out.println(max + "," + min);

      System.out.println(max(5, 8)); // invoke method max()
      System.out.println(min(5, 8)); // invoke method min()
   }

   // The max() method returns the maximum of two given int
   public static int max(int number1, int number2) {
      if (number1 > number2) {
         return number1;
      } else {
         return number2;
      }
   }

   // The min() method returns the minimum of two given int
   public static int min(int number1, int number2) {
      return (number1 < number2) ? number1 : number2;
   }
}
```

return

- Metot gövdesinin içinde, çağırıcıya bir değer döndürmek için bir değer (metodun imzasında bildirilen returnValueType'ın) döndürmek için bir return ifadesi kullanabilirsiniz.

```
return aReturnValue;    // of returnValueType declared in method's signature
return;                 // return nothing (or void)
```

void

- Çağırıcıya bir değer döndürmenize gerek kalmadan belirli eylemleri (ör. Yazdırma) gerçekleştirmek için bir metoda ihtiyacınız olduğunu varsayalım, dönüş değeri türünü void olarak bildirebilirsiniz. Metodun gövdesinde, bir return kullanabilirsiniz; return, kontrolü çağırana döndürmek için dönüş değeri olmayan ifade olarak kullanılabilir. Bu durumda, return ifadesi isteğe bağlıdır. Dönüş ifadesi yoksa, tüm gövde çalıştırılır ve kontrol gövdenin sonunda çağırıcıya tekrar döner.

  Main () 'in dönüş değeri türü void olan bir metot olduğuna dikkat edin. main (), Java çalışma zamanı tarafından çağrılır, gövdede tanımlanan eylemleri gerçekleştirir ve Java çalışma zamanına hiçbir şey geri döndürmez.

```java
import java.util.Scanner;
/**
 * This program contains a boolean method called isMagic(int number), which tests if the
 * given number contains the digit 8.
 */
public class MagicNumber {
   public static void main(String[] args) {
      // Declare variables
      int number;
      Scanner in = new Scanner(System.in);

      // Prompt and read input as "int"
      System.out.print("Enter a positive integer: ");
      number = in.nextInt();

      // Call isMagic() to test the input
      if (isMagic(number)) {
         System.out.println(number + " is a magic number");
      } else {
         System.out.println(number + " is not a magic number");
      }
      in.close();
   }
   public static boolean isMagic(int number) {
      boolean isMagic = false;   // shall change to true if found a digit 8

      // Extract and check each digit
      while (number > 0) {
         int digit = number % 10;   // Extract the last digit
         if (digit == 8) {
            isMagic = true;
            break;  // only need to find one digit 8
         }
         number /= 10;   // Drop the last digit and repeat
      }
      return isMagic;
   }
}
```

```
Enter a positive integer: 1288
1288 is a magic number

Enter a positive integer: 1234567
1234567 is not a magic number
```

Pass-by-Value for Primitive-Type Parameters

- Java'da, ilkel tipte bir argüman bir metoda argüman olarak verildiğinde, bir kopya oluşturulur ve metoda aktarılır. Çağrılan metot klonlanmış kopya üzerinde çalışır ve orijinal kopyayı değiştiremez.
- Hem main () hem de increment () yönteminde number adında bir değişken olmasına rağmen, biri main () 'de, diğeri de increment ()' de bulunan iki farklı kopya aynı ada sahiptir. Programı etkilemeden herhangi birinin adını değiştirebilirsiniz.

```
1  public class PassByValueTest {
2     public static void main(String[] args) {
3        int number = 8, result;
4        System.out.println("In caller, before calling the method, number is: " + number);  // 8
5        result = increment(number); // invoke method with primitive-type parameter
6        System.out.println("In caller, after calling the method, number is: " + number);   // 8
7        System.out.println("The result is " + result);  // 9
8     }
9
10    // Return number + 1
11    public static int increment(int number) {
12       System.out.println("Inside method, before operation, number is " + number); // 8
13       ++number;  // change the parameter
14       System.out.println("Inside method, after operation, number is " + number);  // 9
15       return number;
16    }
17 }
```

a.

```
class ByVal
{
//Ana Program
 public static void main (String[] args)
 {
  int v = 3;
  System.out.println("Yerel değişken ilk değeri: " + v);
  yaz(v);
  System.out.println("Yerel değişken ikinci değeri: " + v);
 }
 //Alt Program
 public static void yaz(int x)
 {
  System.out.println("Parametre ilk değeri: " + x);
  x = 0;     // Parametre değişti
  System.out.println("Parametre ikinci değeri: " + x);
 }
}
```

Programın ekran çıktısı:

```
Yerel değişken ilk değeri: 3
Parametre ilk değeri: 3
Parametre ikinci değeri: 0
Yerel değişken ikinci değeri: 3
```

Pass-by-Reference for Arrays and Objects

- Diziler için, dizi başvurusu metoda aktarılır ve metot, dizi öğelerinin içeriğini değiştirebilir.

```
import java.util.Arrays;  // for Arrays.toString()
public class PassByReferenceTest {
   public static void main(String[] args) {
      int[] testArray = {9, 5, 6, 1, 4};
      System.out.println("In caller, before calling the method, array is: "
            + Arrays.toString(testArray));   // [9, 5, 6, 1, 4]
      // Invoke method with an array parameter
      increment(testArray);
      System.out.println("In caller, after calling the method, array is: "
            + Arrays.toString(testArray));   // [10, 6, 7, 2, 5]
   }

   // Increment each of the element of the given int array
   public static void increment(int[] array) {
      System.out.println("Inside method, before operation, array is "
           + Arrays.toString(array));   // [9, 5, 6, 1, 4]
      // Increment each elements
      for (int i = 0; i < array.length; ++i) ++array[i];
      System.out.println("Inside method, after operation, array is "
           + Arrays.toString(array));   // [10, 6, 7, 2, 5]
   }
}
```

Method Overloading

- Java'da, bir metodun birden fazla sürümü olabilir, her sürüm farklı parametreler kümesi üzerinde çalışır - metot aşırı yükleme olarak bilinir.
- Versiyonlar, parametrelerin sayıları, türleri veya sıralarına göre farklılaştırılacaktır.

```java
/** Testing Method Overloading */
public class AverageMethodOverloading {
   public static void main(String[] args) {
      System.out.println(average(8, 6));     // invoke version 1
      System.out.println(average(8, 6, 9));  // invoke version 2
      System.out.println(average(8.1, 6.1)); // invoke version 3
      System.out.println(average(8, 6.1));
           // int 8 autocast to double 8.0, invoke version 3
      //average(1, 2, 3, 4)  // Compilation Error - no such method
   }

   // Version 1 takes 2 int's
   public static int average(int n1, int n2) {
      System.out.println("version 1");
      return (n1 + n2)/2;  // int
   }

   // Version 2 takes 3 int's
   public static int average(int n1, int n2, int n3) {
      System.out.println("version 2");
      return (n1 + n2 + n3)/3;   // int
   }

   // Version 3 takes 2 doubles
   public static double average(double n1, double n2) {
      System.out.println("version 3");
      return (n1 + n2)/2.0;  // double
   }
}
```

```
version 1
7
version 2
7
version 3
7.1
version 3
7.05
```

Yöntem çağrısı sırasında int'in otomatik olarak double dönüştürülmesinin aksine, int [] double [] olarak dönüştürülmez.

```java
/** Testing Array Method Overloading */
public class SumArrayMethodOverloading {
   public static void main(String[] args) {
      int[] a1 = {9, 1, 2, 6, 5};
      System.out.println(sum(a1));     // invoke version 1
      double[] a2 = {1.1, 2.2, 3.3};
      System.out.println(sum(a2));     // invoke version 2
      float[] a3 = {1.1f, 2.2f, 3.3f};
      //System.out.println(sum(a3));   // error - float[] is not casted to double[]
   }

   // Version 1 takes an int[]
   public static int sum(int[] array) {
      System.out.println("version 1");
      int sum = 0;
      for (int item : array) sum += item;
      return sum;  // int
   }

   // Version 2 takes a double[]
   public static double sum(double[] array) {
      System.out.println("version 2");
      double sum = 0.0;
      for (double item : array) sum += item;
      return sum;  // double
   }
}
```

"boolean" Methods

```
1  /**
2   *  Testing boolean method (method that returns a boolean value)
3   */
4  public class BooleanMethodTest {
5     // This method returns a boolean value
6     public static boolean isOdd(int number) {
7        if (number % 2 == 1) {
8           return true;
9        } else {
10          return false;
11       }
12    }
13
14    public static void main(String[] args) {
15       System.out.println(isOdd(5));  // true
16       System.out.println(isOdd(6));  // false
17       System.out.println(isOdd(-5)); // false
18    }
19 }
```

```
public static boolean isOdd(int number) {
   if (number % 2 == 0) {
      return false;
   } else {
      return true;
   }
}
```

```
public static boolean isEven(int number) {
   return (number % 2 == 0);
}
public static boolean isOdd(int number) {
   return !(number % 2 == 0);
}
```

Matematiksel Yöntemler

```
double Math.pow(double x, double y) // returns x raises to power of y
double Math.sqrt(double x)          // returns the square root of x
double Math.random()                // returns a random number in [0.0, 1.0)
double Math.sin()
double Math.cos()
```

```
Math.PI    // 3.141592653589793
Math.E     // 2.718281828459045
```

```
int secretNumber = (int)Math.random()*100;  // Generate a random int between 0 and 99

double radius = 5.5;
double area = radius*radius*Math.PI;
area = Math.pow(radius, 2)*Math.PI;         // Not as efficient as above

int x1 = 1, y1 = 1, x2 = 2, y2 = 2;
double distance = Math.sqrt((x2-x1)*(x2-x1) + (y2-y1)*(y2-y1));
int dx = x2 - x1;
int dy = y2 - y1;
distance = Math.sqrt(dx*dx + dy*dy);        // Slightly more efficient
```

Rastgele Sayı (Random Number) Üretimi

Hemen hemen tüm programlama dillerinde rastgele sayı üreten bir fonksiyon vardır, bu fonksiyon random ya da rnd ismiyle anılır. Java dilinde rastgele sayı üretimi için ya **Math.random()** yöntemi yada java.util paketinin altındaki **Random** sınıfı kullanılır.

- **Math.random** () yöntemi, 0 ile 1 arasında sayı üretir. Eğer 0 ile 10 arasında sayı üretmek istiyorsak `Math.random ()*10` ifadesini kullanmamız gerekir.
- 0 ile 10 arasında rasgele sayı üretmek için (program başında `import java.util.Random;` deyimi kullanmak koşulu ile) aşağıdaki satırları yazmak yeterlidir:

```
Random rnd = new Random();
sayi = rnd.nextInt(10);
```

Örnek 5.5: 0 ile 20 arasında 20 adet rastgele tamsayı üreten ve bu sayıları ekranda gösteren programı yazınız.

Çözüm:

1. yol: 0 ile 20 arasında rastgele sayı üretmek için gerekli komut satırı `(int)(Math.random() * 20)` şeklindedir.

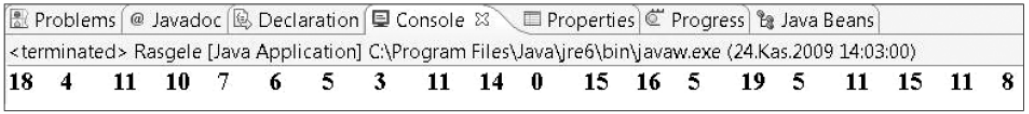
Buna göre programımızı yazarsak:

```
class Rasgele {
public static void main(String args[]) {
       for(int i=0; i < 20; i++) {
       System.out.print((int)(Math.random() * 20)+"\t");
       }
}}
```

2. yol:

```
import java.util.Random;
class Rasgele {
public static void main(String args[]) {
       for(int i=0; i < 20; i++) {
       Random rnd = new Random();
       System.out.print( rnd.nextInt(20)+"\t");
       }
}}
```

Her iki programında herhangi bir andaki ekran çıktısı aşağıdaki gibi olur:



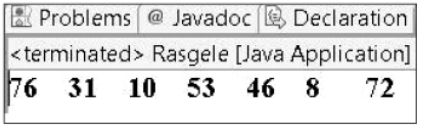
Örnek 5.6: 7 ile 77 arasında 7 adet rastgele tamsayı üreten ve bu sayıları ekranda gösteren programı yazınız.

Çözüm: 7 ile 77 arasında rastgele sayı üretmek için gerekli komut satırı `7+(int)(Math.random() * 70)` şeklindedir.

Buna göre programımızı yazarsak:

```
class Rasgele {
public static void main(String args[]) {
       for(int i=0; i < 7; i++) {
       System.out.print(7+(int)(Math.random() * 70)+"\t");
       }
}}
```

Programın herhangi bir andaki ekran çıktısı:



KAYNAKLAR

- Programming Notes. (2021, March 11). Retrieved from https://www3.ntu.edu.sg/home/ehchua/programming/index.html
- Çobanoğlu B. (2020) Java ile Programlama ve Veri Yapıları. İstanbul Pusula Yayıncılık. 978-605-2359-84-6