Introduction to Java

Methods

مقدمه

- نیاز به نوشتن برنامه‌های فرعی در برنامه زمانی مشاهده می شود، که بخواهیم یک برنامه بزرگ یا نسبتاً بزرگ را پیاده‌سازی کنیم. در اینصورت سعی می‌کنیم، برنامه را به قسمت‌های مجزا و جداگانه از هم تقسیم‌بندی کرده، سپس توسط برنامه‌های فرعی، قطعات جداگانه را پیاده‌سازی کرده و در نهایت آنها را به برنامه اصلی پیوند دهیم. استفاده از برنامه‌های فرعی یکی از اصول برنامه‌نویسی ساخت‌یافته می‌باشد و خوانایی برنامه توسط آنها افزایش می‌یابد.
- برنامه‌های فرعی معمولاً قسمتهای مستقلی از برنامه هستند، که به تنهایی عمل خاصی را انجام می‌دهند. با این ویژگی می‌توان برنامه‌هایی نوشت، که دارای قسمتهای جداگانه و مشخص باشند و هر قسمت یک یا چند وظیفه از وظایف کلی برنامه را به انجام می‌رساند. لذا غالباً برنامه را به قسمت‌های مجزا از هم تقسیم‌بندی می‌کنند و هر قسمت توسط یک روال یا تابع پیاده‌سازی می‌شود و نتایج در برنامه اصلی فراخوانی می‌شوند. از مزایای دیگر استفاده از برنامه های فرعی رفع اشکال سریع برنامه، استفاده بهینه از حافظه، تولید قطعات با قابلیت استفاده مجدد و غیره می‌باشد.

مثال: برنامه ای بنویسید که اطلاعات حداکثر ۱۰۰ دانشجو که عبارتند از:

اسم و فامیلی
شماره دانشجویی
جنسیت
نمره سه تا درس
تعداد واحد هر درس

از ورودی دریافت کرده سپس :
۱- اطلاعات دانشجویان را بر اساس اسم و فامیلی مرتب نماید و لیست حضور غیاب تولید کند (بازای هر درس)
۲- دانشجویان با معدل اول بهمراه مشخصات آنها نمایش دهد.
۳- مشخصات دانشجویان با معدل سوم را نمایش دهد و مشخص کند چه تعدادی هستند.
۴- دانشجویان مشروط و مشخصات آنها را نمایش دهد.
۵- نموداری رسم نماید که در آن فراوانی معدل بر اساس جنسیت در بازه های مشخص نمایش داده شود.

Defining Instance Methods

- Method definitions include a method signature and a method body.
- Methods signatures are defined with the following syntax:

```
modifier return_type method_name(type name, ...)
```

- The return type can be:
  - a fundamental data type
  - an object reference
  - void (no return)
- Parameters are optional
  - If the method takes no parameters, empty brackets are required ()
  - Multiple parameters are separated by commas
  - Parameters are defined by type and name
    - A parameter is a local variable whose scope is the method.

Defining Instance Methods - Visibility

- Methods have the same visibility modifiers as variables
  - public - the method can be invoked from anywhere
  - private - the method can only be invoked from within the class
  - protected - the method can be invoked directly from within the class, within the package, or from within any subclass.
  - default (no modifier specified) - the method can be invoked directly from within the package
- If a method is part of the class's public interface (external view), the method should be public
- If a method is part of the class's internal implementation (ie, support method, etc), it should be private.
- Be careful using default or protected. Use only when justified.

Defining Instance Methods - Body

- A method's body contains all the statements to be executed as part of the method
- The method body is contained within curly braces after the method definition:
  - Use {} placement and indentation to clearly show code structure

```
public class CalculationSheet
{
  public void performCalculations()
  {
       [... method body ...]
  }

  public void clearSheet()
  {
  }
[...]
}
```

Returning values from methods

- A method which has a non-void return type MUST return a value
  - The return value's type must match the type defined in the method's signature.
  - A void method can use a return statement (with no return value) to exit the method.
  - The return value can be used the same as any other expression.

```
public class Car
{
  private int currentGear;
  private int currentRpms;

  public int calculateSpeed()
  {
       return currentRpms * currentGear;
  }
}
```

Passing Parameters to Methods

- Method parameters are declared in the method's signature.
- When a method invocation is made, any parameters included in the invocation are passed to the method
  - All parameters are passed by value. Ie, a copy is made
    - The value of fundamental data types are copied
    - The value of object references (ie memory addresses) are copied
- Parameters become variables within the method. They are not known outside the method.

```
public float calculateInterestForMonth(float rate)
{
  return lowBalanceForMonth * (rate/12.0);
}
```

Overloading Methods

- Java allows for method overloading.
- A Method is overloaded when the class provides several implementations of the same method, but with different parameters
  - The methods have the same name
  - The methods have differing numbers of parameters or different types of parameters
  - The return type MUST be the same

```
public float calculateInterestForMonth()
{
   return lowBalanceForMonth * (defaultRate/12.0);
}

public float calculateInterestForMonth(float rate)
{
   return lowBalanceForMonth * (rate/12.0);
}
```

مثال ساختار تابع

MyClass.java

```java
public class MyClass {
  static void myMethod() {
    System.out.println("I just got executed!");
  }

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

Result:

```
I just got executed!
```

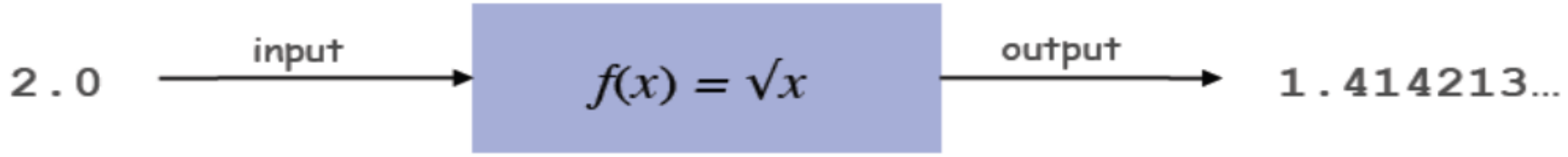
Example

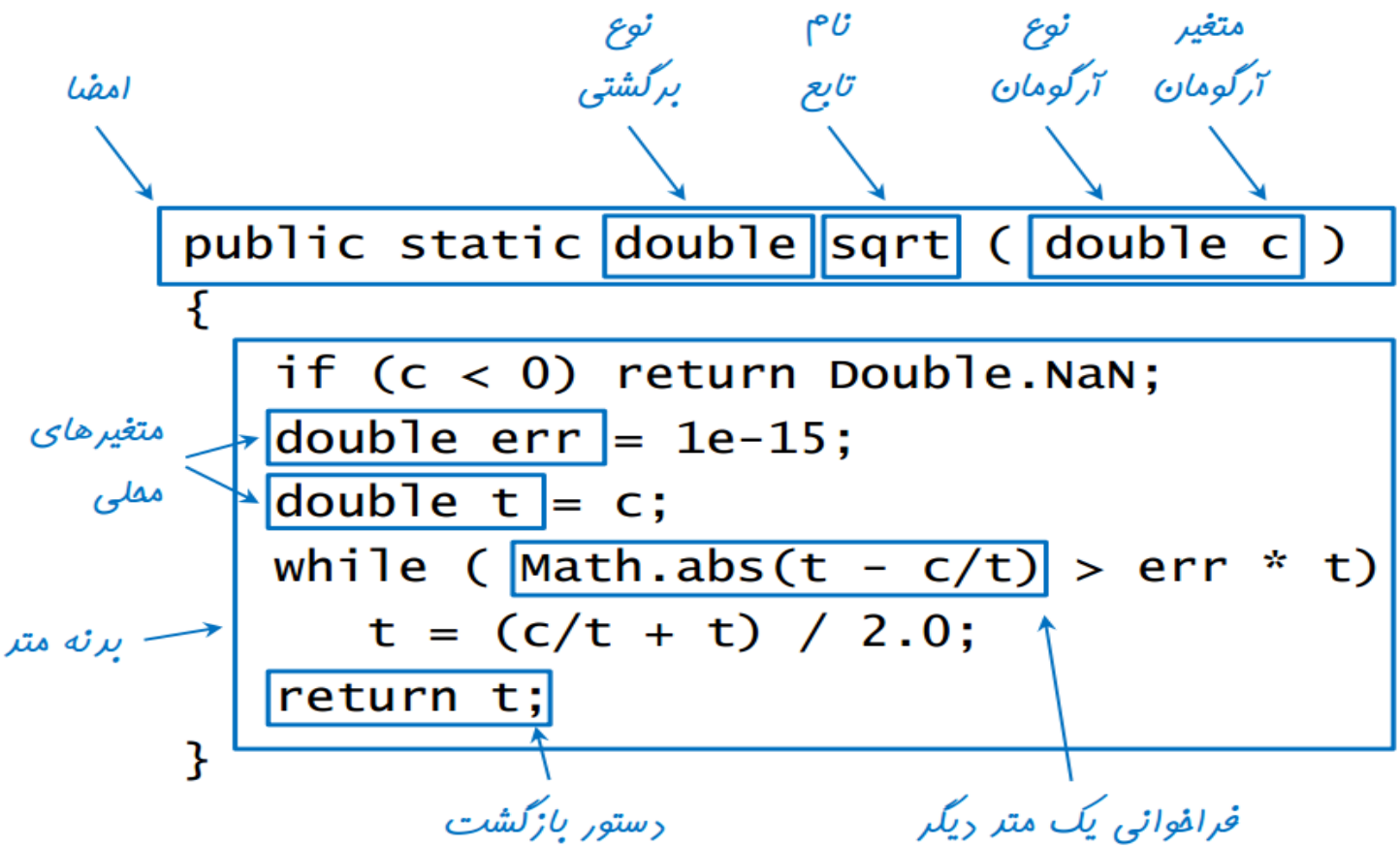
```
public class MyClass {
  static void myMethod() {
    System.out.println("I just got executed!");
  }

  public static void main(String[] args) {
    myMethod();
    myMethod();
    myMethod();
  }
}

// I just got executed!
// I just got executed!
// I just got executed!
```

متدهای با پارامتر

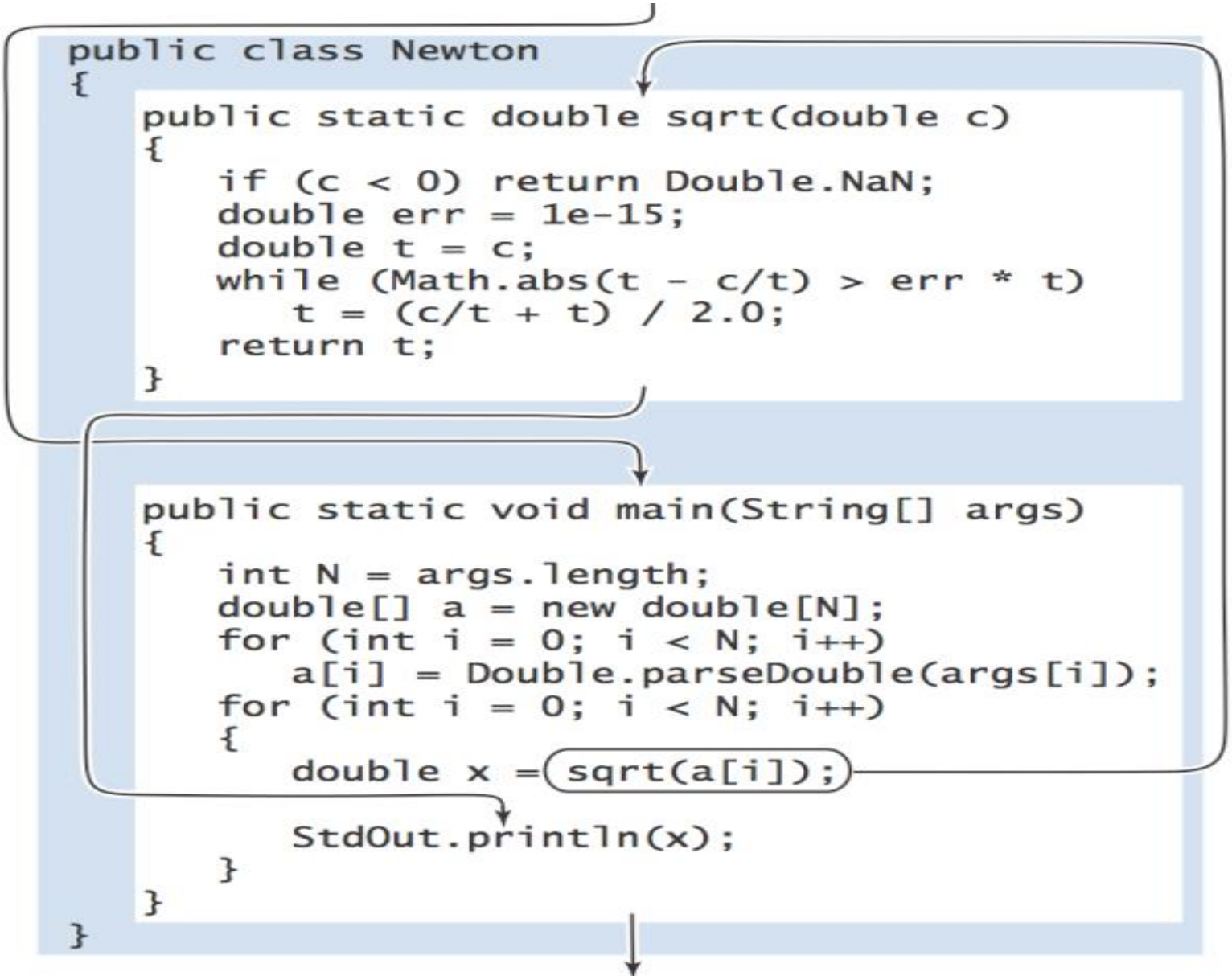
```java
public class MyClass {
  static void myMethod(String fname) {
    System.out.println(fname + " Refsnes");
  }

  public static void main(String[] args) {
    myMethod("Liam");
    myMethod("Jenny");
    myMethod("Anja");
  }
}
// Liam Refsnes
// Jenny Refsnes
// Anja Refsnes
```

ساختار تابع


2.0
input
$f(x) = \sqrt{x}$
output
1.414213...


امضا
نوع برگشتی
نام تابع
نوع آرگومان
متغیر آرگومان
public static double sqrt (double c)
{
if (c < 0) return Double.NaN;
double err = 1e-15;
double t = c;
while (Math.abs(t - c/t) > err * t)
t = (c/t + t) / 2.0;
return t;
}
متغیرهای محلی
بدنه متد
دستور بازگشت
فراخوانی یک متد دیگر

فراخوانی تابع



مثال

```java
public class MyClass {
  static void myMethod(String fname, int age) {
    System.out.println(fname + " is " + age);
  }

  public static void main(String[] args) {
    myMethod("Liam", 5);
    myMethod("Jenny", 8);
    myMethod("Anja", 31);
  }
}

// Liam is 5
// Jenny is 8
// Anja is 31
```

مثال

```
public class MyClass {
  static int myMethod(int x) {
    return 5 + x;
  }

  public static void main(String[] args) {
    System.out.println(myMethod(3));
  }
}
// Outputs 8 (5 + 3)
```

مثال

```
public class MyClass {
  static int myMethod(int x, int y) {
    return x + y;
  }

  public static void main(String[] args) {
    System.out.println(myMethod(5, 3));
  }
}
// Outputs 8 (5 + 3)
```

مثال

MyClass.java

```java
public class MyClass {

  // Create a checkAge() method with an integer parameter called age
  static void checkAge(int age) {

    // If age is less than 18, print "access denied"
    if (age < 18) {
      System.out.println("Access denied - You are not old enough!");

    // If age is greater than 18, print "access granted"
    } else {
      System.out.println("Access granted - You are old enough!");
    }

  }

  public static void main(String[] args) {
    checkAge(20); // Call the checkAge method and pass along an age of 20
  }
}
```

Result:

```
Access granted - You are old enough!
```

مثال

برنامه ای بنویسید که عددی از ورودی دریافت کرده سپس اول بودن، کامل بودن و فیبوناچی بودن را توسط سه متد بررسی نماید.

ارسال پارامترها به عنوان آرگومان

- **Pass by value**

- **Pass by reference**

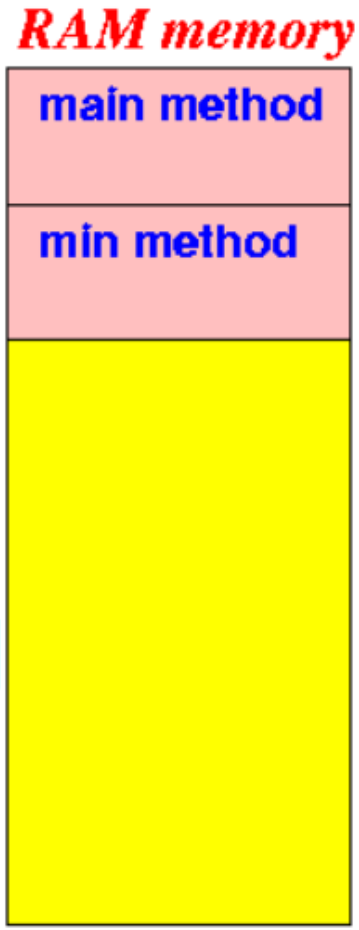
مثال

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

       double x = 1.0, y = 4.0;
       double r;
       r =  ToolBox.fun(  x ,  y );

       System.out.println(x);
       System.out.println(y);
       System.out.println(r);
   }
}
```

?

```
public class ToolBox
{
    public static double  fun  ( double a, double b  )
    {
        double m = 0;
        a = a + 1;
        b = b + 2;
        m = a + b;
        return(m);

    }
}
```

Update parameter variable !



آرایه ها به عنوان آرگومان

- ارسال type= methodName(arrayName)

- دریافت type methodName(arrayType [] arrayName)

مثال

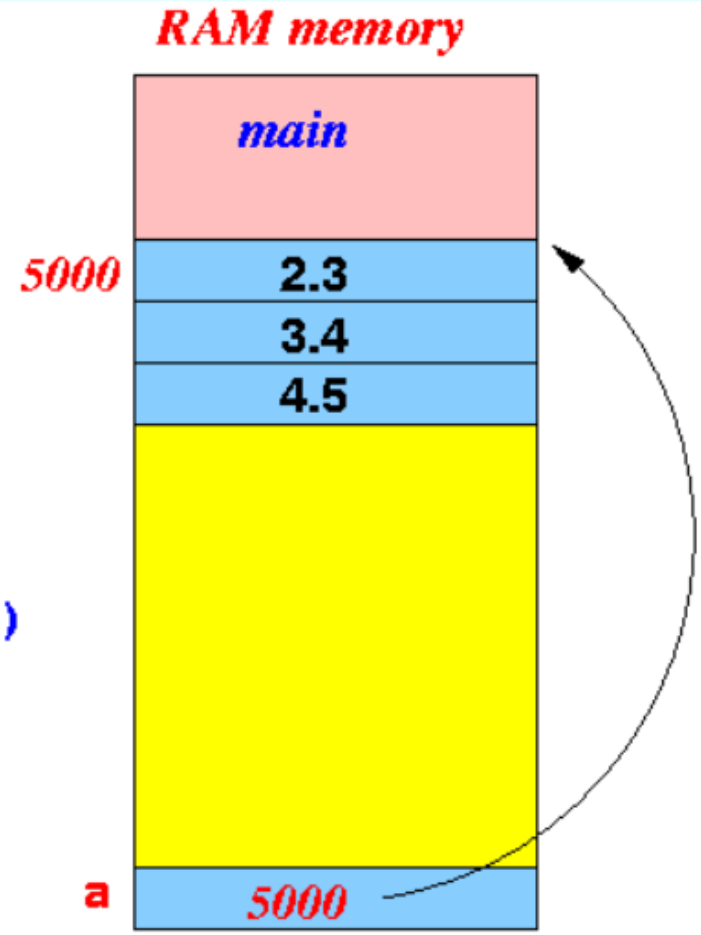
```
public class ArrayParam1
{
   public static double minArray( double[] a )
   {
      int i;            // array index
      double min;       // Current min value

      min = a[0];       // Initial min. value

      for ( i = 1 ; i < a.length ; i++ )
      {
         if ( a[i] < min )
         {
           min = a[i];  // Found a smaller min. value
         }
      }

      return(min);      // Return the min found (instead of printing it)
   }


   public static void main(String[] args)
   {
      double[] a = { 2.3, 3.4 , 4.5, 5.6, 1.2, 7.8, 8.9 }; // 7 elements
      double[] b = { -8.8, 9.7, -14.6, 89.8 };              // 4 elements

      int i;            // Array index
      double min;       // Current min value

      /* ---------------- min of array a ------------------- */

      min = minArray( a ) ;     // Find min value in array a

      System.out.println( min );

      /* ---------------- min of array b ------------------- */

      min = minArray( b ) ;     // Find min value in array b

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

مثال

```
public class ArrayParam2
{
   public static void updateParam( double[] x )
   {
      x[0] = 9999;  // Update one of the array element
   }


   public static void main(String[] args)
   {
      double[] a = { 2.3, 3.4 , 4.5 };

      System.out.println("Array before calling updateParam:");

      for (int i = 0; i < a.length; i++)
         System.out.println( a[i] );

      updateParam( a );        // Call updateParam

      System.out.println("Array AFTER calling updateParam:");

      for (int i = 0; i < a.length; i++)
         System.out.println( a[i] );
   }
}
```

مثال

```
public static void main(String[ ]  args)
{
    ...
    double[ ]  a = {2,3, 3.4, 4.5 };
    updateParam( a );
    ...
}

public static void updateParam( double [ ] x )
{
    x[0] = 9999;
}
```



مثال

```
class sortNumbers
{
    public static void main(String[] args)
    {
        int[] data={40,50,10,30,20,5};
        System.out.println("Unsorted List is :");
        display(data);
        sort(data);
        System.out.println("\nSorted List is :");
        display(data);
    }
    static void display(int num[])
    {
      for(int i=0; i<num.length;i++)
          System.out.print(num[i] + " ");
    }
    static void sort(int num[])
    {
      int i, j, temp;
      for(i=0; i<num.length-i;i++)
          {
             for(j=0; j<num.length-i-1;j++)
                {
                    if(num[j]>num[j+1])
                     {
                         temp = num[j];
                         num[j] = num[j+1];
                         num[j+1] = temp;
                     }
                }
          }
    }
}
```

مثال

find the maximum of the arrays value

```
public static double max(double[] a) {
    double max = Double.NEGATIVE_INFINITY;
    for (int i = 0; i < a.length; i++)
        if (a[i] > max) max = a[i];
    return  max;
}
```

dot product

```
public static double dot(double[] a double[] b) {
    double sum = 0.0;
    for (int i = 0; i < a.length; i++)
        sum += a[i] * b[i];
    return  sum;
}
```

exchange two elements in the array

```
public static void exch(String[] a, int i, int j) {
    String temp = a[i];
    a[i] = a[j];
    a[j] = temp;
}
```

مثال:

برنامه ای بنویسید که دو آرایه مرتب از ورودی دریافت نماید. سپس توسط متدی بنام **merge** دو آرایه را طوری در هم ادغام نماید که آرایه حاصل مرتب باشد. در نهایت آرایه حاصل را در متد اصلی چاپ نماید.

مثال

برنامه ای بنویسید که یک آرایه از ورودی دریافت نماید. سپس با دریافت عددی از ورودی توسط متدی بنام search محل وقوع آن را پیدا کرده در متد اصلی چاپ نماید

مثال

جستجوی دودویی

```
int binarySearch (int [ ] A, int n , int x)
{
        int middle , L , H ;
        L = 0 ;
        H = n-1 ;
        while (L <= H)
        {
                middle = (L+H)/2 ;
                if (x = = A[middle])
                        return (middle +1) ;
                if (x >A[middle])
                        L = middle +1 ;
                else
                        H = middle -1 ;
        }
        return (0) ;
}
```

مثال:

برنامه ای بنویسید که یک آرایه را از ورودی دریافت کرده، سپس با دریافت عددی از

ورودی در صورت وجود عدد را از لسیت حذف نماید.