гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net

Компютърно образователен център Progress Палитра от възможности!

Лекция 4.1

Алгоритми,Елементи на структурно програмиране – управляващи структури

- **Използване на основни техники при програмиране**
- **Пълен формат за синтаксис на командата if**
  Команда за многократно изпълнение на блок от команди – while
- **Задачи**

Разработката на софтуер включва следните основни фази :

- Излседване и анализ на изискванията(събиране и анализиране на реалните изисквания на потребителите)

- Моделиране на архитектура(избор на модел, технология и средства на разработка, които най-пълно отговарят на изискванията)

- Планиране (изработване на план за изпълнение на разработката и описание как този план ще се изпълни, планират се и необходимите ререусри и стойност)

-Изпълнение на софтуера и инсталация при потребителя, поддръжка

Необходимост от изследване на изискванията на потребителя :

- Добре изследвани и анализирани изисквания намалява броя на нежеланиете промени

- Всяка нежелана промяна води до увеличаване на разходите от 5 до 10 пъти на всеки следващ етап от разработката
- Изисква се пълно запознаване с действията, които трябва да изпълнява приложението

- Последователността, в която тези действия трябва да се изпълнят

-Нужда от документиране и визуализация – блок схеми и UML

Стъпки при разработка на приложение :

1.Добийте ясна представа за изискванията на задачата, която програмирате

2. Намерете отговор на следните въпроси :
- Какво точно се иска от програмата да прави?
- Какви са входните данни (брой и тип) и как се иска да стане тяхното въвеждане?
- Какво се извежда като резултат от работата на програмата и в какъв формат
- Има ли още нещо, което програмата трябва да прави

Пример:
Потребителят иска програма за преобразуване на стойностите на температура :
от Fahrenheit в Celsius
от Celsius в Fahrenheit

Потребителят иска да въвежда начална стойност на температурата и да извежда преобразуваната й стойност, посредством диалогов прозорец.

- Какво точно се иска от програмата да прави ?
„Искаме програмата да преобразува стойности от температурата по Fahrenheit в Celsius”

- Какви са входните данни (брой и тип) и как се иска да стане тяхното въвеждане?
*„Искаме да въвеждаме температурата Fahrenheit като **цели числа**,посредством **диалогов прозорец**.”*

- Какво се извежда в резултат от работата на програмата и в какъв формат?
*„Искаме да преобразуваната стойност да е число с плаваща запетая с **два знака след десетичната запетая**, изобразено в **диалогов прозорец**.”*

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net

Алгоритми

Определение :

При зададени входни и изходни данни(краен резултат като изпълнение), алгоритъм наричаме изпълнение на последователност от стъпки, които преобразуват входните до изходните данни или желания краен резултат.

- Алгоритъмът се представя с описание на последовтелността от действия, които трябва да се изпълнят

Пример 1:

Зададени са :

входни данни : N числа

изходни данни : същите N числа, но сортирани

Алгоритъм на сортиране : последователност от стъпки, при изпълнението на които входнните числа се извеждат в сортиран порядък

Пример 2:

Зададени са:

входни данни : 1 цяло число,представляващо стойност по Целзий

изходни данни : 1 число с плаваща запетая, представляващо съответната стойност на въведената температура по Фаренхайт

Алгоритъм за преобразуване на температура от Целзий във Фаренхайт : последователност от стъпки, при изпълнението на които от входното число се пресмята и извежда съответната му стойност по склата на Фаренхайт

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net

Компютърно образователен център Progress Палитра от възможности!

Пример за представяне на алгоритъм

Преобразуване на температура от Целзий във Фаренхайт

Използват се променлива tЦелсиус от тип int и променлива tFahrenheit от double тип

Стъпка 1. Въведи едно цяло число и го присвои на променливата tCelsius

Стъпка 2. Пресметни стойността на израза (1.8 * tCelsius) + 32 и го присвои на променливата tFahrenheit

Стъпка 3. Изведи на печат стойността на променливата tFahrenheit

Стъпка 4. Край на алгоритъма

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net

Структури за управление

Логиката за изпълнение на произволен метод(програма) използва три вида структури за управление :

- **Последователна структура**

  - ✓ Преходът от текущата команда към непосредствено следващата е безусловен – командите се изпълняват една след друга в реда, в който са зададени)

- **Избирателна структура**

  - ✓ **-**Преходът от текущата команда към непосредствено следващата е в зависимост от изпълнение на условие за преход – изборът на следващата команда за изпълнение е в зависимост от удоволетворяване на логическо условие, чиито резултат е **true** или **false**

- Структура за многократно изпълнение

  - ✓ Реализира многократно изпълнение на група от команди,докато е валидно определено логическо условие, чийто резултат е true или false

Последователна структрура – представя се в Java като оператор за присвояване или съобщение до друг обект

Пример :
Преобразуване на температурата от Целзий във Фаренхайт
Реализиран е от последователни структури

-Предаване на управлението(задава следващата инструкция за изпълнение и тя може да не е посредствено следващата по ред)

Обикновенно се изразява с **goto** команда
Пример
Съберете N числа и изведете получената сума

Стъпка 5...
Стъпка 6... Идете на стъпка 3. // goto 3
Стъпка 7....

- изпълнява се стъпка 3, а не посредствено следващата я стъпка 7

Bohm и Jacopini(1970) доказват :
- **goto** команди не са необходими
- **goto** команди водят до логически грешки и трудно читаеми програми
- **goto** команда да **не се използва нито в този курс**, **нито когато и да е**

Всяка програма може да се напише единтвено със следните 3 структури за управление

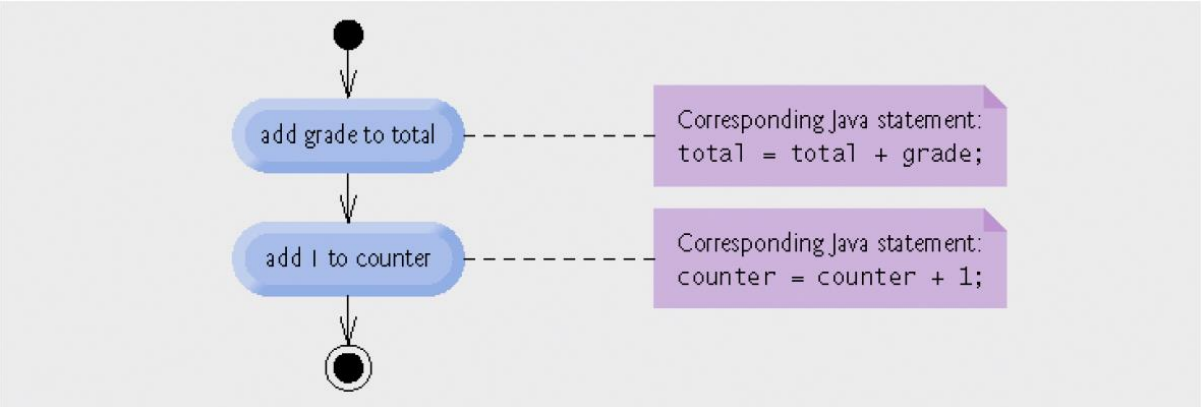
- **Последователна структура**
- **Избирателна структура**
- **Структура за многократно изпълнение**

UML диаграми за дейност(activity diagram)

- Моделират потока от действия на една подсистема от софтуерната разработка
- Използват означение за Действие-Състояние

- Това са **правоъгълници** със заоблени върхове
- **Текст във вътрешността** на тези правоъгълници описва в свободен текст желаното за изпълнение действие
- Предназначени са да представят наличието на последователна структура в програмата
- **Стрелки за преход** определят следващата команда за изпълнение
- **Коментарите** се означават в правоъгълници с подгънат горен десен ъгъл и се свързват към означенията, които коментират с прекъсната линия

ц



гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net

Структури на Java команди, които могат да реализират избирателна структура

- **if** команда (опростен формат)
  - Условие за преход с един изход – когато условието е TRUE, програмата трябва да изпълни една група от команди преди да продължи
- **if….else** команда (стандартен формат)
  - Условие за преход с два изхода(схематично изобразена на предишния екран)
  - - когато условието е **TRUE**, програмата изпълнява една **група от команди** и **друга група от команди** когато условието е **FALSE**, преди да продължи
- **switch** команда
  - Условие за преход с повече от два изхода – аналогично действие
  - Предстои да бъде въведена и демонстрирана на следващите лекции

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net

Java команди, които могат да реализират структура за многократно изпълнение

- Известни като цикли на повторение
- Многократно се изпълнява група от действия(команди),докато зададено условие на цикъла остава TRUE

Видове цикли :

- Предварително **известен брой** на повторения(„въведи 10 числа,”изведи 5 празни реда на печат” – въвеждането се повтаря точно 10 пъти, извеждането на празни редове се повтаря точно 5 пъти)
- Броят на повторенията **не е известен** предварително(„Въведи положително число”,”Да прекратим ли изпълнението?” – не е ясно колко пъти потребителя може да въведе по погрешка отрицателно число или колко пъти ще иска да изпълни програмата, преди да я прекрати

  -Реализира се с **while** команда(повтаря нула или повече пъти група от команди, зададени като тяло на цикъла, докато условието на цикъла има стойност **TRUE**)
  или
  do…**while** команда(повтаря един или повече пъти група от команди, зададени като тяло на цикъла, докато условието на цикъла има стойност TRUE)

Обобщение

Управлението на логиката в методите, принадлежащи на Java класовете се постига с команди, групирани в три вида структури на управление

- Последователна структура

- Избирателна структура

- Структура за многократно изпълнение

Логиката на управление на всеки метод се изразява единствено с тези структури

Тези структури могат да се изпълняват

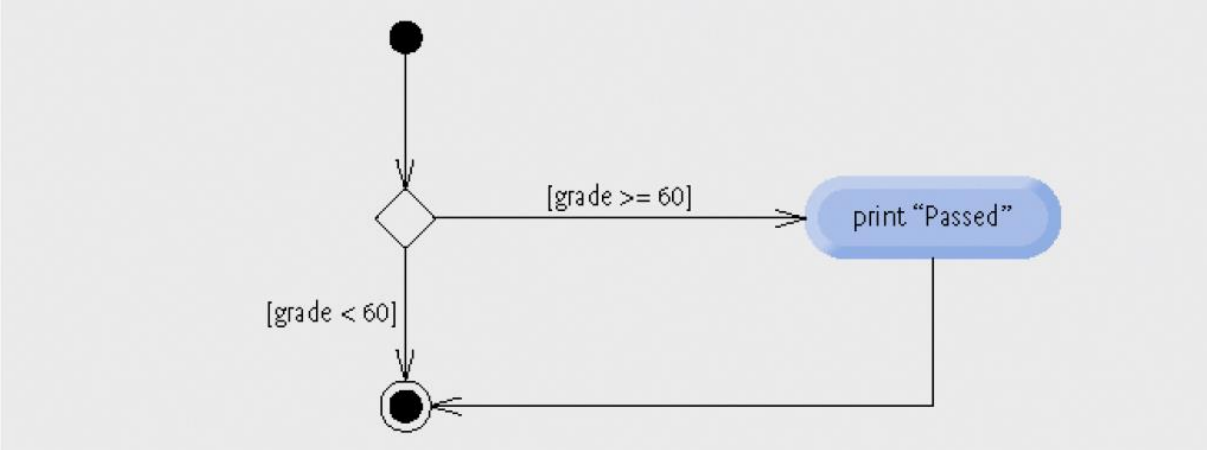
- Последователно една след друга
- Вложени изцяло една в друга

if команда опростен формат

if команда се използва за :
- Да изпълнява група от действия ако условието за преход е true
- Представя се с ромб(diamond) в UML диаграмите
    - Стрелки за преход, излизащи от върховете на ромба указват следващата команда за изпъление в зависимост от условието за преход
    - Състоянието на действието се променя по посока на стрелките, за които съответното им условие има стойност true

```
if ( <условие за преход> )
{
   /* група(блок) от команди, които се изпълняват при резултат TRUE
   на условието за преход */
}
```

UML диаграма на примерно условие за преход



Пример за псевдокод на изобразената if команда
IF student`s grade is geater than or equal to 60 – Print Passed
- Педставя условие за преход „student`s grade is greater than or equal to 60”
- Ако условието за преход е TRUE да се изведе печат „ Passed

Java синтаксис на изобразената if команда :

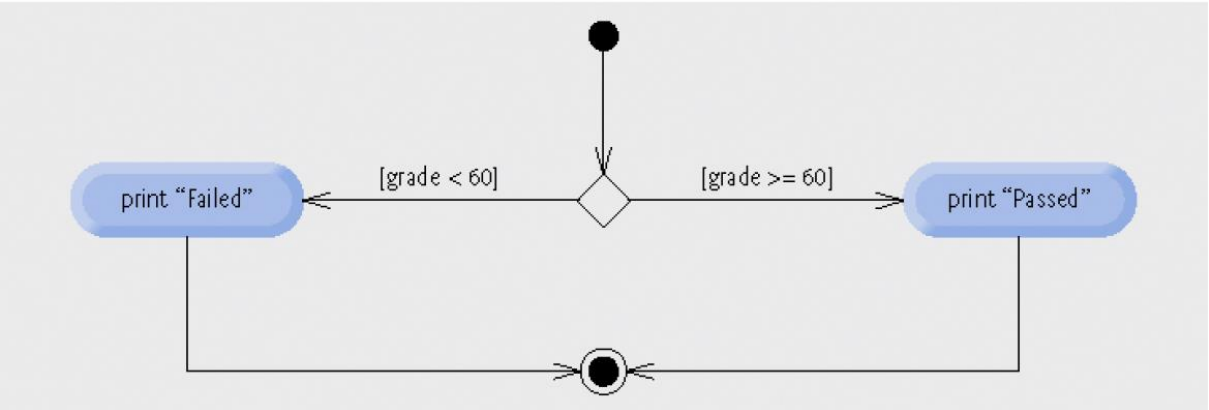
```
if( studentGrade >= 60 )
{
   System.out.println( “Passed” );

}
```

if…else команда с два изхода

- Изпълнява една група от команди, когато условието е true или друга група от команди, когато условието е false, преди да продължи

```
if ( <условие за преход> )
{
   /* група(блок) от команди, които се изпълняват при резултат TRUE
   на условието за преход */
}
else
{
/* група(блок) от команди, които се изпълняват при резултат FALSE
   на условието за преход */
}
```

UML диаграма на примерен if-else statement



- Примери за използване на if-else

Пример за псевдокод на изобразената if команда „

```
If student`s grade is greater than or equal to 60
        Print “Passed”
Else
        Print “Failed”
```

Java синтаксис на изобразената if команда :

```
if( studentGrade >= 60)
{
        System.out.println(“Passed”);
}
else
{
        System.out.println(“Failed”);
}
```

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net

Условен оператор за присвояване – синтаксис

< условие > ? **< стойност,когато условието е TRUE>** : **<стойност, когато условието е FALSE >** ;

- Единствен оператор в Java три операнда
- Използва се вместо if…else, когато на една променлива се присвоява една от две стойности в зависимост от стойността на зададено условие
  - Операторът връща втория операнд, ако първият операнд е true
  - Операторът връща третия операнд, ако първият операнд е false
  - Типично приложение на операторът за условие на присвояване е да присвои абсолютната стойност на дадено число

```
x = ( x < 0 ) ? –x : x;

// Как ще изглежда същото с if…else ?
```

Пример :

```
System.out.println( studentGrade >= 60 ? “Passed” : “Failed” );
```

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net

Вложени if…else команди

- if…else Могат да се влагат в други управляващи структури,респективно други if…else Команди

Псевдокод

```
if student`s grade is greater than or equal to 90
        Print “A”
    else
        If student`s grade is greater than or equal to 80
            Print “B”
        else
            If Student`s grade is greater than or equal to 70
                Print “C”
            else
                If student`s grade is greater than or equal to 60
                    Print “D”
                else
                    Print “F”
```

Java код

```
if ( studentGrade >= 90 )
   System.out.println(“A”);
else
   if ( studentGrade >= 80 )
         System.out.println(“B”);
   else
         if ( studentGrade >= 70 )
               System.out.println(“C”);
         else
               if( studentGrade >= 60)
                     System.out.println( “D”);
               else
                     System.out.println(“F”);

System.out.println( “End of execution” );
```

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net

Блок от код

- Фигурните скоби { } дефинират блок от код
- Групи от команди в if или else се обособяват в блок от код

Случай на „висящ” – else

-Групите от команди след else се свързват с предходния if освен ако не са използвани { }

- Пропускането на { } за ограждане на вложена структура за управление на if …. else води до **логически грешки**

```
// Каква е разликата между
if ( x > 5 )
   if ( y > 5 )
      System.out.println( "x and y are > 5" );
else
      System.out.println( "x is <= 5" );
//  .... и ...
if ( x > 5 )
   if ( y > 5 )
      System.out.println( "x and y are > 5" );
    else
      System.out.println( "x is <= 5" );
// ... И в двата случай има логическа грешка
// При x по- малко от  5 нищо не се разпечатва
```

Случай на „висящ” – else правилно **кодиране**

```
if ( x > 5 )
{
   if(y > 5 )
          System.out.println(“x and y are > 5”);
}else
          System.out.println( “x is <= 5” ) ;
```

Групиране на последователност от команди в блок от код

```
if(  grade >= 60 )
    System.out.println( “Passed” );
else
{
    System.out.println( "Failed”);
    System.out.println(“You must take this course again.”);
}
```

Обичайна грешка при програмиране :

Пропускането на едната или двете фигурни скоби, обособяващи блок от код води до логически и синтактични грешки.

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net

while команда за цикъл

- Позволява многократно изпълнение на група от команди, докато условието на цикъла е true

Пример за псевдо код

While there are more items on my shopping list

Purchase next item and cross it off my list

- **Условие на цикъла**
  - **there are more items on my shopping list**
- **Команди за изпълнение**
  - **Purchase next item and cross it off my list**

while команда синтаксис на Java :

```
while( < условие на цикъла > )
{
   //тяло на цикъла
   // блок от команди за повторно изпълнение
}
```

Пример : Принтирайте числата от 1 до 10, като всяко число е на отделен ред

```
int number = 1;
while( number <= 10 )
{
        System.out.println( number );
```

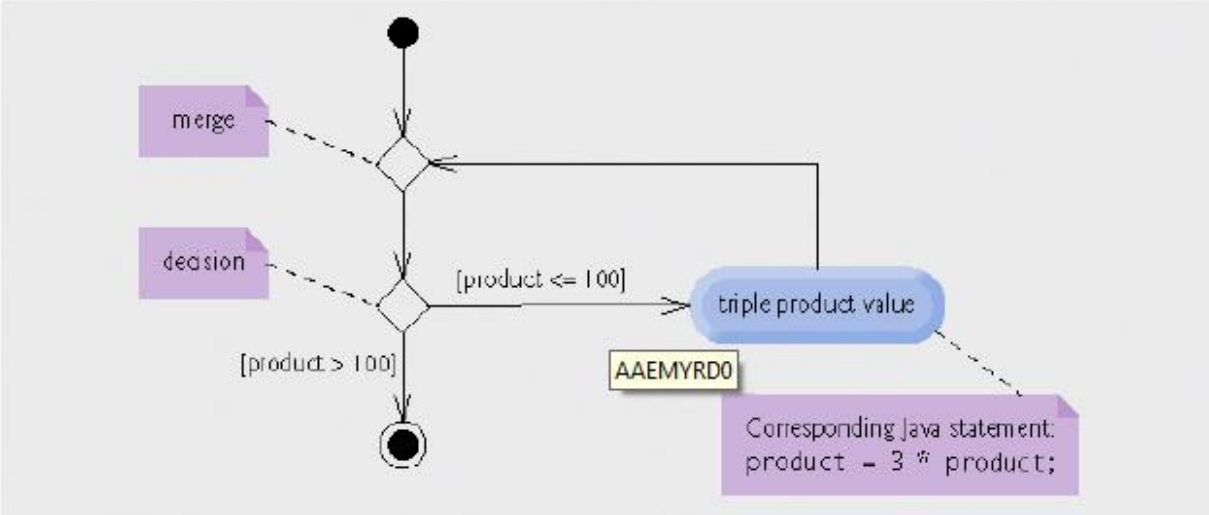

```
        number++;
}
```

Пример : Намерете най-голямата степен на 3, която е по-малка или равна на 100

```
int product = 3;
while( product <= 100 )
{
   product = 3 * product;
}
```

while Команда UML представяне в диаграма на действие

- Използва се символ за смесване – свързва два и повече потока на изпълнение
- Представя се като ромб(аналогично на символа за условен преход) , НО за разлика от символа за условен преход :
    - Има повече от една стрелка за преход, сочеща към върховете на ромба
    - Има единствена изходяща стрелка за преход
    - Няма условия за преход над никоя от стрелките за преход



Обичайна грешка при програмиране :

Пропускането на команда в тялото на цикъла, която води да промени условието за цикъл води до логическа грешка, изразяваща се в безкрайно повторение (infinite loop) и абортиране на програмата.

Дефиниране на алгоритми : Повторение, управлявано от брояч

Повторение, управлявано от брояч

- Използва променлива, която служи да се отброява броят на изпълненията на цикъла, наричат се итерации
- Целочислена променлива

Пример : Ще допълним class GradeBook от последната лекция за решаване на следната задача

- Една група от **10 студента** се е явила на изпит и са известни получените оценки.Това са **цели числа в интервала от 0 до 100**.Трябва да се направи програма за пресмятане на средната оценка на групата студенти.

```
1   Инициализирай променливата total на 0
2    Инициализирай променливата gradeCounter на 1
3
4   While gradeCounter е по- малко или равно на 10
5           Извести потребителя да въведе следващата оценка
6           Въведи следващата оценка и я присвои на nextGrade
7           Добави nextGrade към променливата total
8            Добави 1 към gradeCounter
9
10  Присвои на променливата classAverage стойността на израза total/ 10
10
11  Изведи на печат променливата classAverage
```

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author Vankov
 */
import java.util.Scanner;

public class GradeBook {

    private String courseName;//name of course which GradeBook represents

    public GradeBook(String name) {
        setCourseName(name);//store the course name

    }

    public void setCourseName(String name) {
        if (name != null) {//validation for String data
            courseName = name;
        }
    }

    public String getCourseName() {
        return courseName;
    }

    public void displayMessage() {
        System.out.printf("Welcome to the gradebook for \n%s!\n\n",
                getCourseName());
    }

```

.... →

```
35
36      public void determineClassAverage() {
37
38          Scanner input = new Scanner(System.in);
39
40          int total;//used for sum of all entered grades
41          int gradeCounter;//counter of grades
42          int grade;//grade value which user has entered
43          int average;//average for all grades
44
45          total = 0;
46          gradeCounter = 1;
47
48          while (gradeCounter <= 10) {
49
50              System.out.print("Enter grade: ");
51              grade = input.nextInt();
52              total += grade;
53
54              gradeCounter++;//counter must be incremented on every step of executic
55          }
56          average = total / 10;//
57          //display results
58          System.out.printf("Sum of all grades is %d\n", total);
59          System.out.printf("Class average is %d\n", average);
60
61
62      }
63  }
64
```

Правило за добро програмиране :

Съставянето на коректен алгоритъм е най-важната част от разсработването на дадена програма.

```
5
6   /**
7    *
8    * @author Vankov
9    */
10  public class GradeBookTest {
11
12      public static void main(String args[]){
13
14          //Constructor which passes name of course is called
15          GradeBook myGradeBook =new GradeBook("Introduction to java programming"+
16                  " with Computer education center \"Progress\"");
17
18          myGradeBook.displayMessage();
19          myGradeBook.determineClassAverage();
20
21      }
22
23  }
24
```

OUTPUT:

```
run:
Welcome to the gradebook for
Introduction to java programmingwith Computer education center "Progress"!

Enter grade: 50
Enter grade: 50
Enter grade: 50
Enter grade: 50
Enter grade: 50
Enter grade: 100
Enter grade: 100
Enter grade: 100
Enter grade: 100
Enter grade: 100
Sum of all grades is 750
Class average is 75
BUILD SUCCESSFUL (total time: 29 seconds)
```

- Въпроси?

гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net

Задачи

1.**Изведете числата от 0 до 40 през 2,разделени със запетая чрез while цикъл.**

очакван резултат : 0,2,4,6,8,10...38,40

2.**Напишете програма, която въвежда цяло число N, което указва бройката на числа, които ще се въвеждат.След това се въвеждат N на брой числа и се изкарва тяхната сума**.

Примерен резултат :

Въведете N: 3

5

10

-5

The sum is : 10

3. **Задача за намиране на най-голямата стойност(т.е. максимумът между група от стойности) се използва често в програмирането. Напишете метод на Java,който реализира следния алгоритъм с три(локални) променливи:**

- counter : брояч на итерациите до 10(следи да се въведат точно 10 цели числа)
- number : съхранява текущото цяло число, въведено от потребителя.
- largest : най-голямото число,въведено до момента

След изпълнение на цикъла извеждаме най-голямата стойност.