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 5.1

Алгоритми с цикли,псевдокод и визуализирането му – 2-ра част

- **Елементи на структурното програмиране – управляващи структури:**
- **Команди за цикъл : for(){ } и do{ }while() .**
- **Програмна реализация на структура за избор с повече от два варианта - switch**
- **Задачи**

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

office@progressbg.net www.progressbg.net

Въведение

- Въвеждаме специална Java команда за алгоритъм за повторение с предварително зададен брой на повторенията – командата for
- Изследваме основните елементи, необходими за програмната реализация на алгоритъм за повторение, с предварително зададен брой на повторенията
- Въвеждаме допълнителна Java команда за алгоритъм за повторение с предварително неизвестен брой на повторенията – do…while
- Въвеждаме специална Java команда за реализация на избирателна структура при повече от два варианта на избор – switch

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!

Градивни елементи на цикли, управлявани от брояч

Реализация на алгоритъм за повторение с предварително зададен брой на повторенията
Тази реализация изисква :
- **Управляваща променлива**(брояч на повторенията/циклите/итерациите)
- Задаване на **начална стойност** за управляващата променлива
- Задаване на **нарастване/намаляване** на управляващата променлива при всеки цикъл
- **Условие за край** на повторенията – проверява дали управляващата променлива е достигнала крайната си стойност

```
1  // Fig. 5.1: WhileCounter.java
2  // Counter-controlled repetition with the while repetition statement.
3
4  public class WhileCounter
5  {
6     public static void main( String args[] )
7     {
8        int counter = 1; // declare and initialize control variable
9
10       while ( counter <= 10 ) // loop-continuation condition
11       {
12          System.out.printf( "%d  ", counter );
13          ++counter; // increment control variable by 1
14       } // end while
15
16       System.out.println(); // output a newline
17    } // end main
18 } // end class WhileCounter
```

```
1  2  3  4  5  6  7  8  9  10
```

Името на управляващата променлива е `counter`. Началната стойност на `counter` е 1

Условие за проверка достигане на крайната стойност на `counter`

Нарастване на `counter` с 1

Обичайни грешки при програмиране :

- Понеже числата с плаваща запетая се съхраняват с определена точност в паметта, то управляващи променливи от тип плаваща запетая могат да доведат до неочаквани и неточни крайни резултати на пресмятане в резултат на изпълнение на по-малък или по-голям брой на повторения от зададения брой.

- Използвайте целочислени променливи за управление на повторенията.

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!

for команда за цикъл с брояч

- for командата е специално създадена за изпълнение на повторения при предварително зададен брой на повторенията
- while команда се използва основно за изпълнение на повторения при предварително известен брой на повторенията
- Управляващата променлива може да описва предварително зададен:
  - Непрекъснат интервал от цели числа
  - Крайно множество от стойности или обекти(в следващите лекции)

ForCounter.java :

```
1  // Fig. 5.2: ForCounter.java
2  // Counter-controlled repetition with the for repetition statement.
3
4  public class ForCounter
5  {
6     public static void main( String args[] )
7     {
8        // for statement header includes initialization,
9        // loop-continuation condition and increment
10       for ( int counter = 1; counter <= 10; counter++ )
11          System.out.printf( "%d  ", counter );
12
13       System.out.println(); // output a newline
14    } // end main
15 } // end class ForCounter
```

```
1  2  3  4  5  6  7  8  9  10
```

Нарастване на брояча `counter`

Проверка на условие за край на цикъла

Управляващата променлива е `counter`

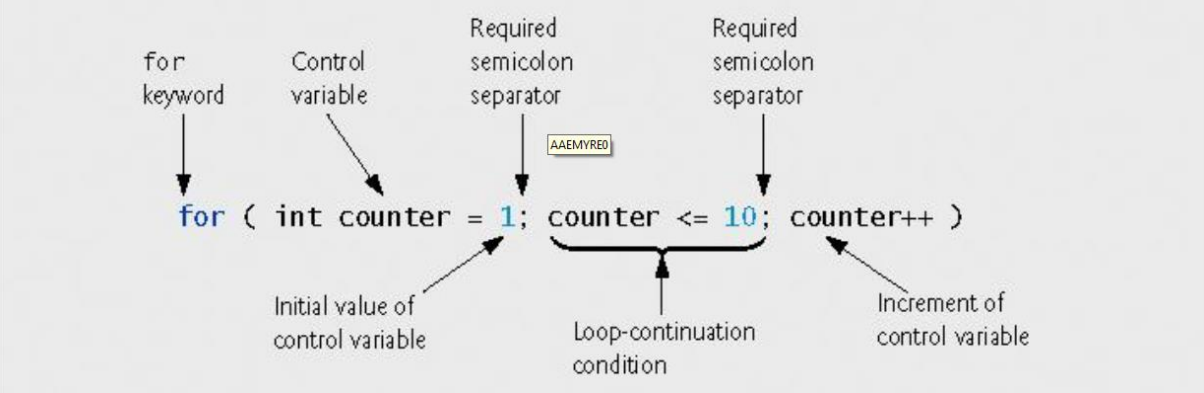
Инициализирана е на 1

При изпълнението на for командата :

1.Изпълнява се частта за инициализиране
2.Проверява се условието за край на цикъла(counter <= 10)
3.Тялото на цикъла се изпълнява, понеже началната стойност на брояча е 1
4.Брояча нараства с 1(counter++) след приключване на последната команда в тялото на цикъла
5.Отново се проверява условието на цикъла и се взима решение за повтаряне на цикъла при новата стойност на брояча или прекратяване на цикъла
6.След извършване на определения брой итерации се изпълнява първата команда ,непосредствено следваща тялото на цикъла

Обичайна грешка при програмиране :
Когато променливата за управление на повторенията е декларирана на мястото на нейното инициализиране в командата, то тя не може да се използва извън тялото на цикъла.

Диаграма на съставните части на for цикъл :




```
for ( инициализация ; условие за край ; нарастване или намаляване )
{
   // тяло на цикъла
}
```

Обичайна грешка при програмиране

Безкраен цикъл се получава, когато условието за край не става никога **false**.

Пример :

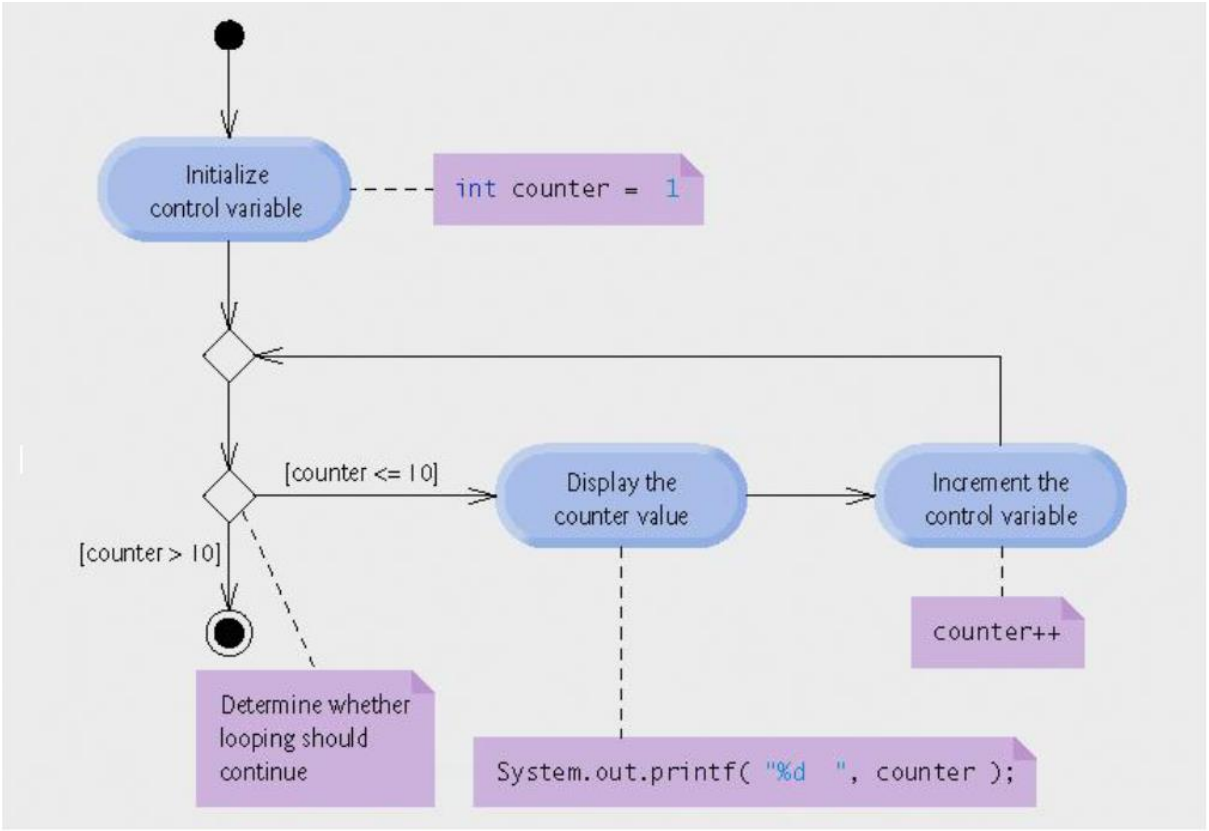
```
for ( int i=0 ; i < 10 ; i--){
//изпълнява се безброй пъти

}
```

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

office@progressbg.net www.progressbg.net

UML диаграма за действие на fig 5.2



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

office@progressbg.net www.progressbg.net

Задаване на нарастването/ намаляването на брояча в for команда

- **Нарастване на брояча от 1 до 100 със стъпка 1**

```
for ( int i = 1; i <= 100; i++ )
```

- **Намаляване на брояча от 100 до 1 със стъпка 1**

```
for ( int i = 100; i >= 1; i-- )
```

- **Нарастване на брояча от 7 до 77 със стъпка 7**

```
for ( int i = 7; i <= 77; i += 7 )
```

- **Намаляване на брояча от 20 до 2 със стъпка 2**

```
for ( int i = 20; i >= 2; i -= 2 )
```

- **Изменение на брояча за последователността: 2, 5, 8, 11, 14, 17, 20**

```
for ( int i = 2; i <= 20; i += 3 )
```

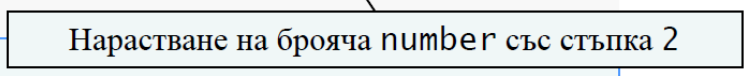
- **Изменение на брояча за последователността : 99, 88, 77, 66, 55, 44, 33, 22, 11, 0**

```
for ( int i = 99; i >= 0; i -= 11 )
```

Sum.java :

```
1  // Fig. 5.5: Sum.java
2  // Summing integers with the for statement.
3
4  public class Sum
5  {
6     public static void main( String args[] )
7     {
8        int total = 0; // initialize total
9
10       // total even integers from 2 through 20
11       for ( int number = 2; number <= 20; number += 2 ) {
12          total += number;
13       }
14       System.out.printf( "Sum is %d\n", total ); // display results
15    } // end main
16 } // end class Sum
```

```
Sum is 110
```



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

office@progressbg.net www.progressbg.net

Примерна задача за използване на for Команда

Пример:

Да се пресметне натрупаната лихва за даден депозит.Нека са депозирани $1,000 в спестовен влог с 5% годишна лихва.При условие,че цялата лихва остава по депозита да се пресметне и изведе цялата парична сума в края на всяка от следващите 10 години след началото на депозита.Използваме следната формула за пресмятане на натрупаната сума :

$$a= p (1+ r)^{n}$$

където :

p – е началната сума на депозита

r – е годишната лихва (използваме 0.05 за да укажем 5% при пресмятане)

n – означава години

a – е сумата ,натрупана по депозита в края на n-тата година

Решението на задачата изисква цикъл за повторно пресмятане на зададената формула за всяка от 10-те години


```java
// Fig. 5.6: Interest.java
// Compound-interest calculations with for.

public class Interest
{
   public static void main( String args[] )
   {
      double amount; // amount on deposit at end of each year
      double principal = 1000.0; // initial amount before interest
      double rate = 0.05; // interest rate

      // display headers
      System.out.printf( "%s%20s\n", "Year", "Amount on deposit" );

      // calculate amount on deposit for each of ten years
      for ( int year = 1; year <= 10; year++ )
      {
         // calculate new amount for specified year
         amount = principal * Math.pow( 1.0 + rate, year );

         // display the year and the amount
         System.out.printf( "%4d%,20.2f\n", year, amount );
      } // end for
   } // end main
} // end class Interest
```

Използваме тип double

Вторият низ е подравнен в дясно и се извежда в поле с дължина от 20 символа

Пресмятаме amount с for команда

Използваме запетая (,) във форматния спесификатор за разделяне на хилядите в цялата част на числото

```
Year   Amount on deposit
   1            1,050.00
   2            1,102.50
   3            1,157.63
   4            1,215.51
   5            1,276.28
   6            1,340.10
   7            1,407.10
   8            1,477.46
   9            1,551.33
  10            1,628.89
```

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

office@progressbg.net www.progressbg.net

Форматиращи спецификатори – допълнителни опции

- Дължина на поле
- Извеждане на (-) форматиращ флаг (%-.2f)
- Извеждане на (+) форматиращ флаг (% +.2f)
- Извеждане на запетая(,) за разделяне на хилядните в цялата част(%,20.2f)

static метод

- клас метод – общ за всички обекти от даден клас

- ClassName.methodName(arguments)

Пример :

Math.pow(x, y)

Пресмята стойността на x^y

Важно : аргументите x и y трябва да са от тип double!

- class Math принадлежи на java.lang
- Съдържа математически библиотечни функции

За повече информация :

http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html

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

office@progressbg.net www.progressbg.net

do…while команда за повторение

- do…while имат логика и изпълнение **аналогично на командата while**
- Използва се за реализиране на алгоритми с повторение, за които броят на повторенията **не е зададен предварително**

- Различава се по мястото за проверка за край на повторенията
- Проверката е след тялото на цикъла,така тялото на цикъла се изпълнява **поне веднъж**

DoWhileTest.java :

```
1  // Fig. 5.7: DoWhileTest.java
2  // do...while repetition statement.
3
4  public class DoWhileTest
5  {
6     public static void main( String args[] )
7     {
8        int counter = 1; // initialize cou
9
10       do
11       {
12          System.out.printf( "%d  ", counter );
13          ++counter;
14       } while ( counter <= 10 ); // end do...while
15
16       System.out.println(); // outputs a newline
17    } // end main
18 } // end class DoWhileTest
```

```
1  2  3  4  5  6  7  8  9  10
```

Декларира и инициализира брояча `counter`

Брояча `counter` се извежда преди проверката за край

Outline

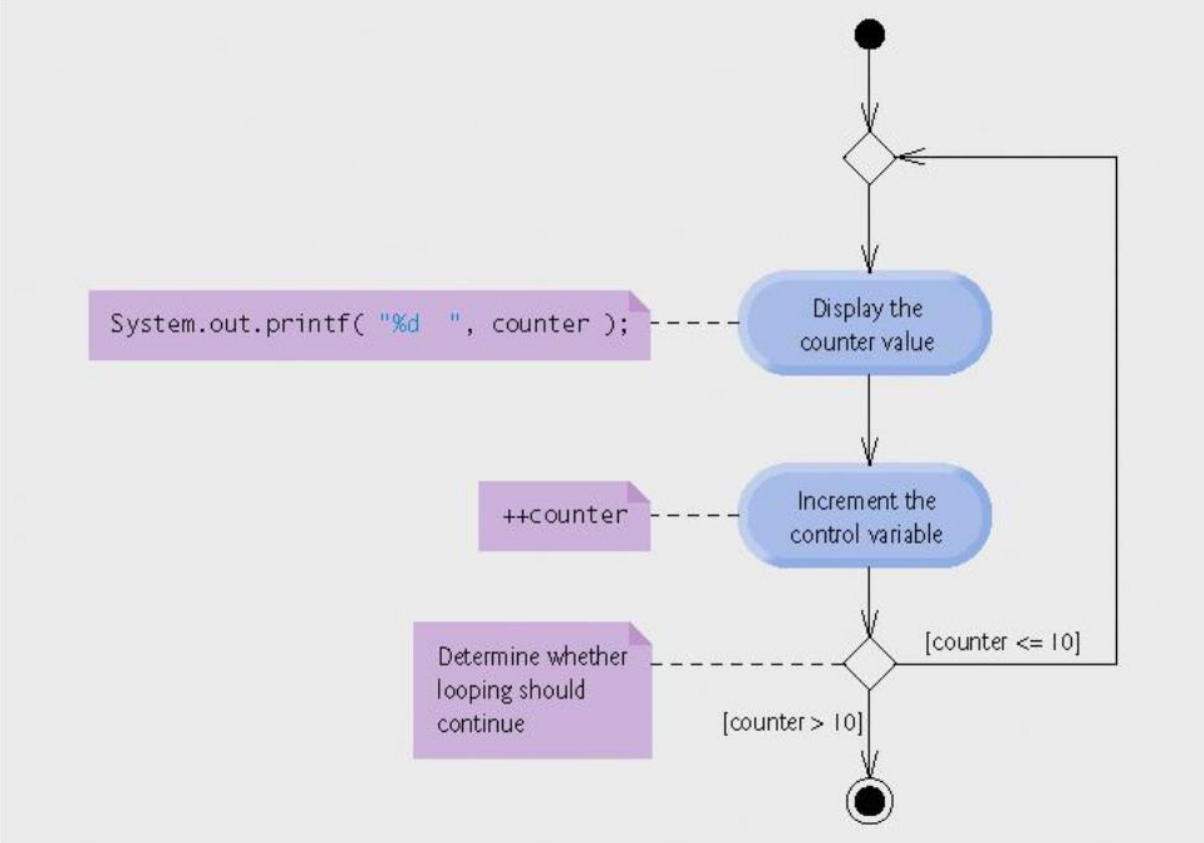
DoWhileTest.java

ред 8

редове 10-14

Program output

UML диаграма за действие на DoWhileTest.java :



Правило за добро програмиране :

Винаги използвайте фигурни скоби за маркиране на тялото на do…while цикъла.

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

office@progressbg.net www.progressbg.net

switch команда – условие за преход с избор при повече от два изхода

- switch команда – реализира условие за избор с повече от два възможни изхода

Пример :
Разглеждаме допълнение към class GradeBook представен лекция 3 и доработен в лекция 4.
Тук в допълнение на пресмятането на средната оценка ще преброим колко от оценките са A,B,C,D или F чрез увеличаване с 1 на съответния брояч

```java
// Fig. 5.9: GradeBook.java
// GradeBook class uses switch statement to count A, B, C, D and F grades.
import java.util.Scanner; // program uses class Scanner

public class GradeBook
{
   private String courseName; // name of course this GradeBook represents
   private int total; // sum of grades
   private int gradeCounter; // number of grades entered
   private int aCount; // count of A grades
   private int bCount; // count of B grades
   private int cCount; // count of C grades
   private int dCount; // count of D grades
   private int fCount; // count of F grades

   // constructor initializes courseName;
   // int instance variables are initialized to 0 by default
   public GradeBook( String name )
   {
      courseName = name; // initializes courseName
   } // end constructor

   // method to set the course name
   public void setCourseName( String name )
   {
      courseName = name; // store the course name
   } // end method setCourseName

```

```
29    // method to retrieve the course name
30    public String getCourseName()
31    {
32       return courseName;
33    } // end method getCourseName
34
35    // display a welcome message to the GradeBook user
36    public void displayMessage()
37    {
38       // getCourseName gets the name of the course
39       System.out.printf( "Welcome to the grade book for\n%s!\n\n",
40          getCourseName() );
41    } // end method displayMessage
42
43    // input arbitrary number of grades from user
44    public void inputGrades()
45    {
46       Scanner input = new Scanner( System.in );
47
48       int grade; // grade entered by user
49
50       System.out.printf( "%s\n%s\n   %s\n   %s\n",
51          "Enter the integer grades in the range 0-100.",
52          "Type the end-of-file indicator to terminate input:",
53          "On UNIX/Linux/Mac OS X type <ctrl> d then press Enter",
54          "On Windows type <ctrl> z then press Enter" );
55
56       // loop until user enters the end-of-file indicator
57       while ( input.hasNext() )
58       {
59          grade = input.nextInt(); // read
60          total += grade; // add grade to
61          ++gradeCounter; // increment numb
62
63          // call method to increment appropriate counter
64          incrementLetterGradeCounter( grade );
65       } // end while
66    } // end method inputGrades
67
68    // add 1 to appropriate counter for specified grade
69    public void incrementLetterGradeCounter( int numericGrade )
70    {
71       // determine which grade was entered
72       switch ( grade / 10 )
73       {
74          case 9:  // grade was between 90
75          case 10: // and 100
76             ++aCount; // increment aCount
77             break; // necessary to exit switch
78
79          case 8: // grade was between 80 and 89
80             ++bCount; // increment bCount
81             break; // exit switch
82
```

Извеждане на промпт съобщение

Условието за край използва `hasNext` за да определи дали има оценки за въвеждане

(grade / 10) е условието за избор

`switch` командата определя кой `case` етикет да се изпълни, в зависимост от условието за избор

```
83          case 7: // grade was between 70 and 79
84             ++cCount; // increment cCount
85             break; // exit switch
86
87          case 6: // grade was between 60 and 69
88             ++dCount; // increment dCount
89             break; // exit switch
90
91          default: // grade was less than 60
92             ++fCount; // increment fCount
93             break; // optional; will exit switch anyway
94       } // end switch
95    } // end method incrementLetterGradeCounter
96
97    // display a report based on the grades entered by user
98    public void displayGradeReport()
99    {
100      System.out.println( "\nGrade Report:" );
101
102      // if user entered at least one grade...
103      if ( gradeCounter != 0 )
104      {
105         // calculate average of all grades entered
106         double average = (double) total / gradeCounter;
107
108         // output summary of results
109         System.out.printf( "Total of the %d grades entered is %d\n",
110            gradeCounter, total );
111         System.out.printf( "Class average is %.2f\n", average );
112         System.out.printf( "%s\n%s%d\n%s%d\n%s%d\n%s%d\n%s%d\n",
113            "Number of students who received each grade:",
114            "A: ", aCount,   // display number of A grades
115            "B: ", bCount,   // display number of B grades
116            "C: ", cCount,   // display number of C grades
117            "D: ", dCount,   // display number of D grades
118            "F: ", fCount ); // display number of F grades
119      } // end if
120      else // no grades were entered, so output appropriate message
121         System.out.println( "No grades were entered" );
122   } // end method displayGradeReport
123} // end class GradeBook
```

default case за оценки под 60

- ✓ Комбинацията от клавиши, маркиращи края на стандартен вход, са зависими от ОС.Обикновено това е Ctrl+z за windows и Ctrl+d за UNIX.

Обичайна грешка при програмиране

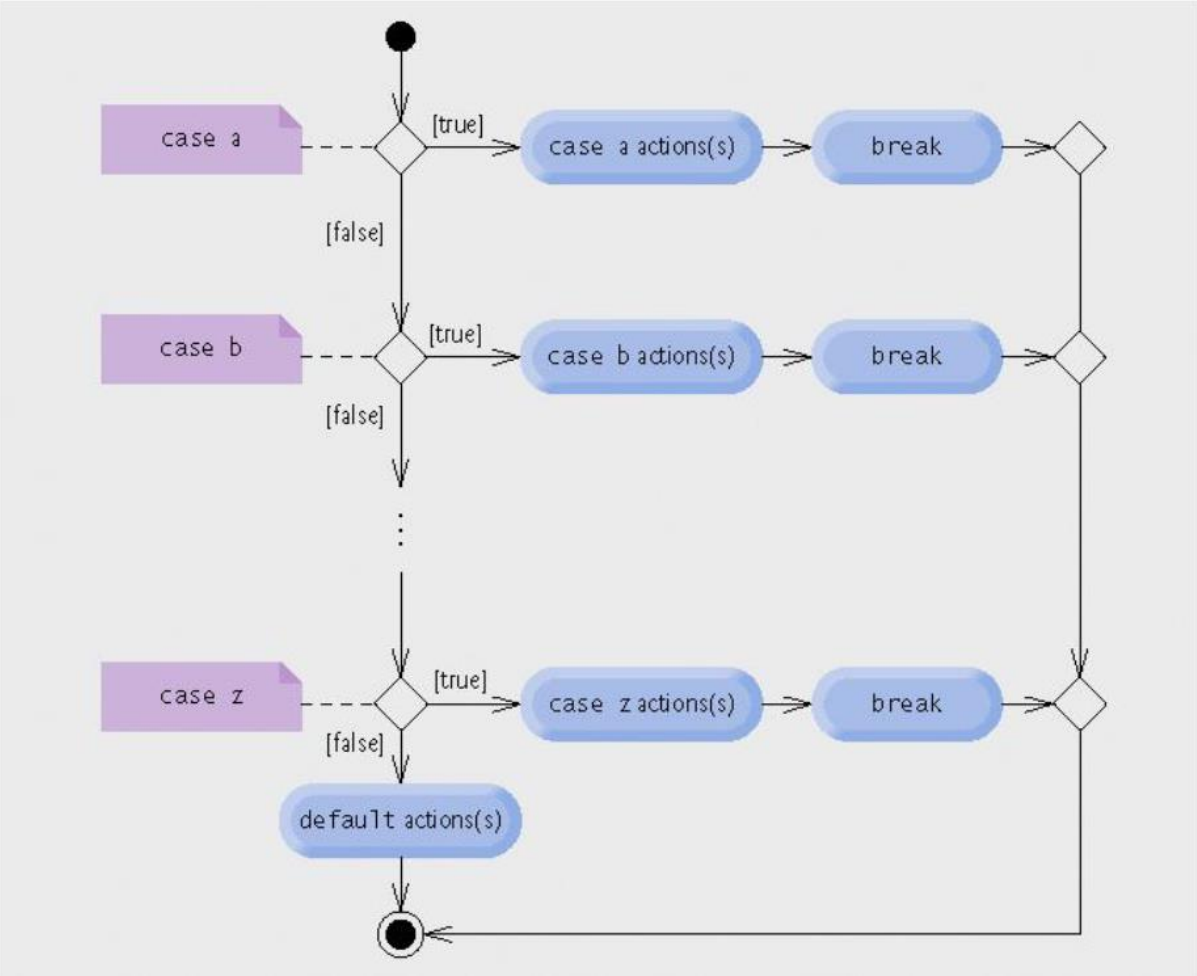
Пропускането на **break** Команда в края на **case** на switch команда е логическа грешка.

GradeBookTest.java :

```
1  // Fig. 5.10: GradeBookTest.java
2  // Create GradeBook object, input grades and display grade report.
3
4  public class GradeBookTest
5  {
6     public static void main( String args[] )
7     {
8        // create GradeBook object myGradeBook and
9        // pass course name to constructor
10       GradeBook myGradeBook = new GradeBook(
11          "CS101 Introduction to Java Programming" );
12
13       myGradeBook.displayMessage(); // display welcome message
14       myGradeBook.inputGrades(); // read grades from user
15       myGradeBook.displayGradeReport(); // display report based on grades
16    } // end main
17 } // end class GradeBookTest
```

Извикване на GradeBook public методи за броене на оценките

UML диаграма на switch команда :



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

office@progressbg.net www.progressbg.net

Задачи

1)Напишете програма, която прочита 5 числа между 1 и 30.
За всяко така прочетено число програмата да извежда един ред от същия брой съседни звездички ,колкото е числото.Например, ако въведеното число е 7, то изведете ред ******* .

2) Какво се извежда в резултат от следните команди ?

```
for ( i = 1; i <= 5; i++ )
{
    for ( j = 1; j <= 3; j++ )
    {
        for ( k = 1; k <= 4; k++ )
            System.out.print( '*' );
            System.out.println();
    } // end inner for
    System.out.println();
} // end outer for
```