

Лекция 4.2

Основни алгоритмични постановки, видове цикли.Използване на графичен потребителски интерфейс(GUI) за създаване на прости графични изображения

- Цикли без предварително зададен брой на повторенията
- Първоначално запознаване с графичния контекст на Java – рисуване на прости рисунки
- Идентифициране на данните на клас
- Задачи

Дефиниране на алгоритми : цикли без предварително известен брой повторения

Краят на циклите се определя от прочитане на подходящо дефиниран **ограничител**

- Броят на повторенията не е предварително известен
- За ограничител(преграда или флаг) се използва знак, низ от символи или друга подходящо дефинирана стойност
- За такъв ограничител може да се избира всяка стойност, която не е допустима, т.е. не е валидна като входна данна
- Пример : Въвеждат се в цикъл цени на стоки. За прекъсване на такъв цикъл да се използва ограничител(sentinel) произволно отрицателно число, т.е. цикълът прекъсва, когато потребителя въведе число по-малко от нула

Фази на дефиниране на алгоритъм

Най-често в изпълнението на една програма се обособяват логически три фази:

- **Фаза на създаване и инициализация** на локалните променливи
- **Фаза за обработка на данните** – въвеждане и обработка
- **Фаза на приключване на изпълнението** – показване(връщане) на крайния резултат и освобождаване на използваните ресурси

Обичайна грешка при програмиране :

При целочислено деление, проверявайте дали не делите на нула и изведете съобщение за грешка,когато установите делене на нула,вместо да оставите програмата да прекъсне.

Псевдокод на алгоритъм за пресмятане на средна оценка със **ограничител**

```
1      Initialize total to zero
2      Initialize counter to zero
3
4      Prompt the user to enter the first grade
5      Input the first grade (possibly the sentinel)
6
7      While the user has not yet entered the sentinel
8          Add this grade into the running total
9          Add one to the grade counter
10         Prompt the user to enter the next grade
11         Input the next grade (possibly the sentinel)
12
13     If the counter is not equal to zero
14         Set the average to the total divided by the counter
15         Print the average
16     else
17         Print "No grades were entered"
```

- ✓ **Прекъсваме да разбиваме(разлагаме)** алгоритъма на елементарни операции по подхода „отгоре – надолу“, когато всяко от описаните в алгоритъма действие е достатъчно ясно да се представи като команда или множество от команди в Java.
- ✓ Понякога опитни програмисти пишат програми без да използват псевдокод или UML представяне на действията в програмата. Това може да е приемливо за прости и добре познати задачи, но е **недопустимо за големи и сложни проекти**.

```

1 // Fig. 4.9: GradeBook.java
2 // GradeBook class that solves class-average program using
3 // sentinel-controlled repetition.
4 import java.util.Scanner; // program uses class Scanner
5
6 public class GradeBook
7 {
8     private String courseName; // name of course this GradeBook represents
9
10    // constructor initializes courseName
11    public GradeBook( String name )
12    {
13        courseName = name; // initializes courseName
14    } // end constructor
15
16    // method to set the course name
17    public void setCourseName( String name )
18    {
19        courseName = name; // store the course name
20    } // end method setCourseName
21
22    // method to retrieve the course name
23    public String getCourseName()
24    {
25        return courseName;
26    } // end method getCourseName
27
28    // display a welcome message to the GradeBook user
29    public void displayMessage()
30    {
31        // getCourseName gets the name of the course
32        System.out.printf( "Welcome to the grade book for\n%s!\n\n",
33            getCourseName() );
34    } // end method displayMessage
35
36    // determine the average of an arbitrary number of grades
37    public void determineClassAverage()
38    {
39        // create Scanner to obtain input from command window
40        Scanner input = new Scanner( System.in );
41
42        int total; // sum of grades
43        int gradeCounter; // number of grades entered
44        int grade; // grade value
45        double average; // number with decimal point for average
46
47        // initialization phase
48        total = 0; // initialize total
49        gradeCounter = 0; // initialize loop counter
50
51        // processing phase
52        // prompt for input and read grade from user
53        System.out.print( "Enter grade or -1 to quit: " );
54        grade = input.nextInt();
55

```

Assign a value to instance variable `courseName`

Declare method `setCourseName`

Declare method `getCourseName`

Declare method `displayMessage`

Декларирай `determineClassAverage`

Декларирай и инициализирай `Scanner` променлива `input`

Декларирай локални `int` променливи `total`, `gradeCounter` и `grade`, както и `double` променлива `average`

..→

```
56 // loop until sentinel value read from user
57 while ( grade != -1 )
58 {
59     total = total + grade; // add grade to total
60     gradeCounter = gradeCounter + 1; // increment counter
61
62     // prompt for input and read next grade from user
63     System.out.print( "Enter grade or -1 to quit: " );
64     grade = input.nextInt();
65 } // end while
66
67 // termination phase
68 // if user entered at least one grade...
69 if ( gradeCounter != 0 )
70 {
71     // calculate average of all grades entered
72     average = (double) total / gradeCounter;
73
74     // display total and average (with two digits of precision)
75     System.out.printf( "\nTotal of the %d grades entered is %d\n",
76         gradeCounter, total );
77     System.out.printf( "Class average is %.2f\n", average );
78 } // end if
79 else // no grades were entered, so output appropriate message
80     System.out.println( "No grades were entered" );
81 } // end method determineClassAverage
82
83 } // end class GradeBook
```

while цикъла се повтаря,
докато grade е
различно от -1
което се явява
ограничител (sentinel),

Пресметни average
като се използва
(double) за явно
преобразуване на
типа

Изведи average grade
закръглено до 2
знака след
запетаята

Изведи съобщение "No grades were
entered"

Сравнение между програмата с ограничител и програмата с повторение управлявано от брояч :

- При повторение, управлявано от брояч всяка итерация на **while** чете поредната стойност от общия брой стойности, зададени от потребителя
- При повторение, управлявано от ограничител, програмата прочита първата стойност **ПРЕДИ** започване на while цикъла. Тази стойност определя дали тялото на цикъла ще се изпълни **поне веднъж**.
- Ако условието на while е **false**, потребителя е въвел ограничителя и тялото на цикъла няма да се изпълни
- Ако обаче, това условие е **true**, тялото на цикъла се изпълнява, добавя се въведената оценка към **total**. После се въвежда следващата оценка и щом се стигне до края на блока от код за тялото на цикъла, то изпълнението прожължава с нова проверка на **while** условието.
- while условието се пресмята с последната оценка от изпит, въведена от потребителя и така се проверява дали тялото на цикъла да се изпълни отново.
- Стойността на променливата **grade** винаги (при този алгоритъм) се въвежда преди проверката за while условието. Това позволява да се провери веднага след въвеждането дали е прочетена валидна стойност или ограничител. При прочитане на ограничител в тази последователност **не се увеличава стойността на брояча gradecounter**.

Правило за добро програмиране :

- При повторение, управлявано от ограничител е необходима в промпта за въвеждане на данни да се упоменава приетата за ограничител стойност.

Обичайна грешка при програмиране :

- Пропускането на скоби при обособяване на блок от команди може да доведе до логически грешки, като например безкраен цикъл. За да го предпазим, тялото на всяка структура е препоръчително да е обособено като блок от команди, дори и да има само една команда в тялото на структурата.

Оператор за явно преобразуване на тип

- Създава временно копие на операнда от типа, който указва

Пример :

(double) създава временно копие на операнда си от тип плаваща запетая с двойна точност

- Нарича се явно преобразуване на тип
double average = total / gradeCounter; // грешно, защото води до загуба на точност
double average = (double) total / gradeCounter; // правилно

Неявно преобразуване на тип

- Директно преобразуване при пресмятания в друг тип (например от **int** към **double**)

double x = 3 + 2.0; // 3 се преобразува неявно до 3.0

- **Неявно преобразуване**, използва се при преобразуване на примитивни данни, заемащи по-малко памет към тип данни заемащи повече памет. За преобразуване в обратна посока е необходимо **явно преобразуване**.
- Например :
byte b = (byte) 2.0 ; // **явно преобразуване**
double d = 2.0 f ; // **неявно преобразуване**

Тип данна	Валидно неявно преобразуване до
double	None
float	double
long	float or double
int	long, float or double
char	int, long, float or double
short	int, long, float or double
byte	short, int, long, float or double
boolean	None (boolean values are not considered to be numbers in Java)

Fig. 4.1 Допустими неявни преобразувания.

```
1 // Fig. 4.10: GradeBookTest.java
2 // Create GradeBook object and invoke its determineClassAverage method.
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.determineClassAverage(); // find average of grades
15    } // end main
16
17 } // end class GradeBookTest
```

Create a new **GradeBook** object

Pass the course's name to the **GradeBook** constructor as a **string**

Call **GradeBook's** **determineClassAverage** method

```
Welcome to the grade book for
CS101 Introduction to Java Programming!
```

```
Enter grade or -1 to quit: 97
Enter grade or -1 to quit: 88
Enter grade or -1 to quit: 72
Enter grade or -1 to quit: -1
```

```
Total of the 3 grades entered is 257
Class average is 85.67
```

Вложени управляващи структури

Ще дефинираме алгоритъма „отгоре – надолу“, последователно уточняване на стъпките

Ще покажем, как отделни управляващи структури

Могат да се влагат една в друга, чрез поместването им в тяло от команди на обхващата ги структура.

Задача :

Един колеж предлага курс за лицензиране на брокери. През изминалата година 10 студента са положили изпит по този курс.

Колежът иска да знае колко от студентите са взели успешно изпита и трябва да се напише програма за обобщаване на резултатите.

Налице е списък с имена на 10 студента като срещу всяко име е отбелязано 1, ако студентът е взел изпита и 2, ако студентът не е взел изпита.

Дадено е, че програмата трябва да анализира данните по следния начин:

1. Въвежда се всеки от резултатите от изпита (т.е., 1 или 2). Изписва съобщение “Enter result” (Въведете резултат) на екрана като промпт за въвеждане на данните

2. Преброява броя на изпитните резултати от всеки тип (издържали и неиздържали изпита).

3. Извежда обобщение на резултатите, показващи броя на издържалите и броя на неиздържалите изпита.

4. Ако повече от 8 студента са издържали изпита, изведи съобщение „Raise tuition“ (Увеличете таксата)

Псевдокод :

```
1  Initialize passes to zero
2  Initialize failures to zero
3  Initialize student counter to one
4
5  While student counter is less than or equal to 10
6    Prompt the user to enter the next exam result
7    Input the next exam result
8
9    If the student passed
10     Add one to passes
11  Else
12     Add one to failures
13
14  Add one to student counter
15
16 Print the number of passes
17 Print the number of failures
18
19 If more than eight students passed
20  Print "Raise tuition"
```

Analysis.java :

```
1 // Fig. 4.12: Analysis.java
2 // Analysis of examination results.
3 import java.util.Scanner; // class uses class Scanner
4
5 public class Analysis
6 {
7     public void processExamResults
8     {
9         // create Scanner to obtain input from command window
10        Scanner input = new Scanner( System.in );
11
12        // initializing variables in declarations
13        int passes = 0; // number of passes
14        int failures = 0; // number of failures
15        int studentCounter = 1; // student counter
16        int result; // one exam result (obtains value from user)
17
18        // process 10 students using counter-controlled loop
19        while ( studentCounter <= 10 )
20        {
21            // prompt user for input and obtain value from user
22            System.out.print( "Enter result (1 = pass, 2 = fail): " );
23            result = input.nextInt();
24
25            // if...else nested in while
26            if ( result == 1 )
27                passes = passes + 1; // increment passes;
28            else
29                failures = failures + 1; // increment failures
30
31            // increment studentCounter so loop eventually terminates
32            studentCounter = studentCounter + 1;
33        } // end while
34
35        // termination phase; prepare and display results
36        System.out.printf( "Passed: %d\nFailed: %d\n", passes, failures );
37
38        // determine whether more than 8 students passed
39        if ( passes > 8 )
40            System.out.println( "Raise Tuition" );
41    } // end method processExamResults
42
43 } // end class Analysis
```

Деклариране на локални
променливи за метод
`processExamResults()`

`while` цикъла продължава
докато
`studentCounter <= 10`

Определя дали студента е изкарал и
увеличава брояча за анализирани
вече студенти

Пресмята дали има повече от 8
студента изкарали изпита

AnalysisTest.java :

```
1 // Fig. 4.13: AnalysisTest.java
2 // Test program for class Analysis.
3
4 public class AnalysisTest
5 {
6     public static void main( String args[] )
7     {
8         Analysis application = new Analysis(); // create Analysis object
9         application.processExamResults(); // call method to process results
10    } // end main
11
12 } // end class AnalysisTest
```

Създаване на Analysis
обект

```
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Passed: 9
Failed: 1
Raise Tuition
```

Повече от 8 студента с анзкарали изпита

```
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Passed: 6
Failed: 4
```

Обичайна грешка при програмиране :

Инициализиране на локалните променливи при тяхната декларация, позволява да се избегне евентуална грешка при компилация от използване на неинициализирани променливи.

Increment и decrement

Оператор	Наречен	Пример	Обяснение
++	префикс increment	++a	Увеличава a с 1 , после използва новата стойност на a в израза, в който е поместен.
++	постфикс increment	a++	Използва текущата стойност на a , в който е поместен, после увеличава a с 1 .
--	префикс decrement	--b	Намалява b с 1 , после използва новата стойност на b в израза, в който е поместен.
--	постфикс decrement	b--	Използва текущата стойност на b , в който е поместен, после намалява b с 1 .

```

1 // Fig. 4.16: Increment.java
2 // Prefix increment and postfix increment operators.
3
4 public class Increment
5 {
6     public static void main( String args[] )
7     {
8         int c;
9
10        // demonstrate postfix increment operator
11        c = 5; // assign 5 to c
12        System.out.println( c ); // print 5
13        System.out.println( c++ ); // print 5 then postincrement
14        System.out.println( c ); // print 6
15
16        System.out.println(); // skip a line
17
18        // demonstrate prefix increment operator
19        c = 5; // assign 5 to c
20        System.out.println( c ); // print 5
21        System.out.println( ++c ); // preincrement then print 6
22        System.out.println( c ); // print 6
23
24    } // end main
25
26 } // end class Increment

```

Post incrementing the c variable

Preincrementing the c variable

```

5
5
6

5
6
6

```

GUI и Graphics : Прости рисунки на Java

Java координатна система

- Дефинирана е с x-координати и y-координати
- x – по хоризонталата и y по вертикалата
- Координатите се нанасят по x – ос и y – ос
- Координатите единици са пиксели, а група от x и y представляват точка(point).

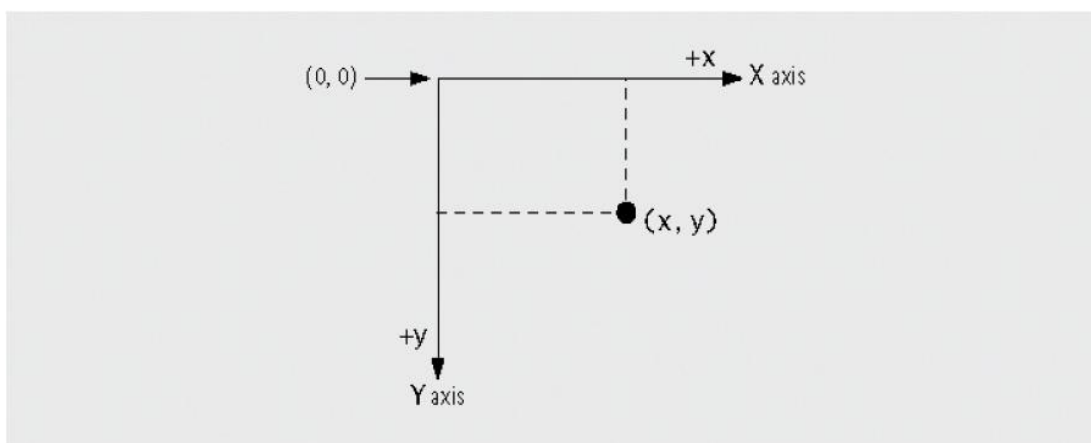
class **JPanel** е в библиотека **javax.swing** package

- Предоставя област върху която може да се рисува или добавят компоненти

class **Graphics** е в библиотека **java.awt** package

- Предоставя методи за рисуване на форми, текст и други

Java coordinate system(Java координатна система) :



Въведение в онаследяването

- Базов(**суперклас**) и производен клас (**субклас**)
 - Производния клас използва ключовата дума **extends**
 - Производния клас онаследява от базовия клас:
 - Всички данни и методи на базовия клас, които не са **скрити** за него.
 - Производния клас разширява базовия клас
- Пример : Библиотечния class **JPanel** има методи
paintComponent()
getWidth()
getHeight()
- То всеки производен на него клас ще може да използва същите тези методи като част от своята дефиниция

DrawPanel.java :

```
1 // Fig. 4.19: DrawPanel.java
2 // Draws two crossing lines on a panel.
3 import java.awt.Graphics;
4 import javax.swing.JPanel;
5
6 public class DrawPanel extends JPanel
7 {
8     // draws an X from the corners of the panel
9     public void paintComponent( Graphics g )
10    {
11        // call paintComponent to ensure the panel displays correctly
12        super.paintComponent( g );
13
14        int width = getWidth(); // total width
15        int height = getHeight(); // total height
16
17        // draw a line from the upper-left to the lower-right
18        g.drawLine( 0, 0, width, height );
19
20        // draw a line from the lower-left to the upper-right
21        g.drawLine( 0, height, width, 0 );
22    } // end method paintComponent
23 } // end class DrawPanel
```

Import the `java.awt.Graphics` and
the `javax.swing.JPanel` classes

class `DrawPanel` разширява
class `JPanel`

Декларира метод `paintComponent()`

Извлича ширината и
височината на `JPanel`'s

Рисува две линии

class **JPanel** – Swing компонент, името започва с ‘J’

- Всеки обект **JPanel** има метод **paintComponent()**
 - **paintComponent()** се извиква автоматично при рисуване на обекта **JPanel**
 - Задължително се декларира като
`public void paintComponent(Graphics g){}`
 - Аргументът Graphics g се предоставя при изпълнението на **paintComponent()**
 - Първата команда в **paintComponent()** трябва да е `super.paintComponent(g);`
 - Всеки обект на class **JPanel** има методи **getWidth()** и **getHeight()** – връщат съответно ширина и височина на **JPanel** като цели числа.

За пълен набор от методи и описание на класа :

<http://docs.oracle.com/javase/7/docs/api/javax/swing/JPanel.html>

- Всеки class **Graphics** обект има метод **drawLine(int x1, int y1, int x2, int y2)**
 - Чертае линия от точката зададени с първите два аргумента(x1,y1) до точката, зададена с вторите два аргумента(x2,y2)

За пълен набор от методи и описание на класа :

<http://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html>

class **JFrame** принадлежи на **javax.swing** библиотека

- Служи за изобразяване на прозорец (window)
- Метод **setDefaultCloseOperation()**
- Приема аргументи указващи събитие при натискане на бутона за затваряне (x), използва **JFrame.EXIT_ON_CLOSE** като аргумент указващ на приложението да прекрати, когато потребителя затвори прозореца
- Метод **add()** – Добавя обект от даден тип компонент към прозореца **JFrame**

- Метод `setSize()` – Задава широчина и височина на текущия `JFrame`, те се подават като целочислени аргументи

За пълен набор от методи и описание на класа:

<http://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html>

```
1 // Fig. 4.20: DrawPanelTest.java
2 // Application to display a DrawPanel.
3 import javax.swing.JFrame;
4
5 public class DrawPanelTest
6 {
7     public static void main( String args[] )
8     {
9         // create a panel that contains our drawing
10        DrawPanel panel = new DrawPanel();
11
12        // create a new frame to hold the panel
13        JFrame application = new JFrame();
14
15        // set the frame to exit when it is closed
16        application.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
17
18        application.add( panel ); // add the panel to the frame
19        application.setSize( 250, 250 ); // set the size of the frame
20        application.setVisible( true ); // make the frame visible
21    } // end main
22 } // end class DrawPanelTest
```

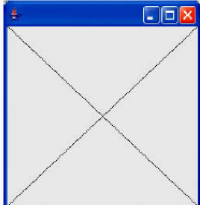
Import на `JFrame` class от библиотека `javax.swing`

Създава съответно `DrawPanel` и `JFrame` обекти

Задава приключване на приложението при затваряне на прозореца

Добави `DrawPanel` обект към `JFrame` обекта (`application`)

Задай размер на прозореца и покажи прозореца (обекта `application` от class `JFrame`)



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

office@progressbg.net

www.progressbg.net

