

Лекция 2.1

Въведение в ООП
Принципи на ОО
Дизайн.
Първа програма
на Java.

Основни теми

- **Въведение**
- **Въведение в обектно ориентираното програмиране.**
- **Представяне на обекти и класове.**
- **Атрибути и стойности.**
- **Действия и съобщения**
- **Въведение в синтаксиса на Java**
- **Обобщение**
- **Задачи**

- 2.1** Взаимодействия между обекти
- 2.2** Представяне на обекти
- 2.3** Атрибути и стойности
- 2.4** Действия и съобщения
- 2.5** Структура на клас- променливи и методи
- 2.6** Променливи и методи на клас и обект
- 2.7** Първа програма на Java: Печат на ред от текст
- 2.8** Изпълнение на програмата
- 2.9** Обобщение
- 2.10** Задачи

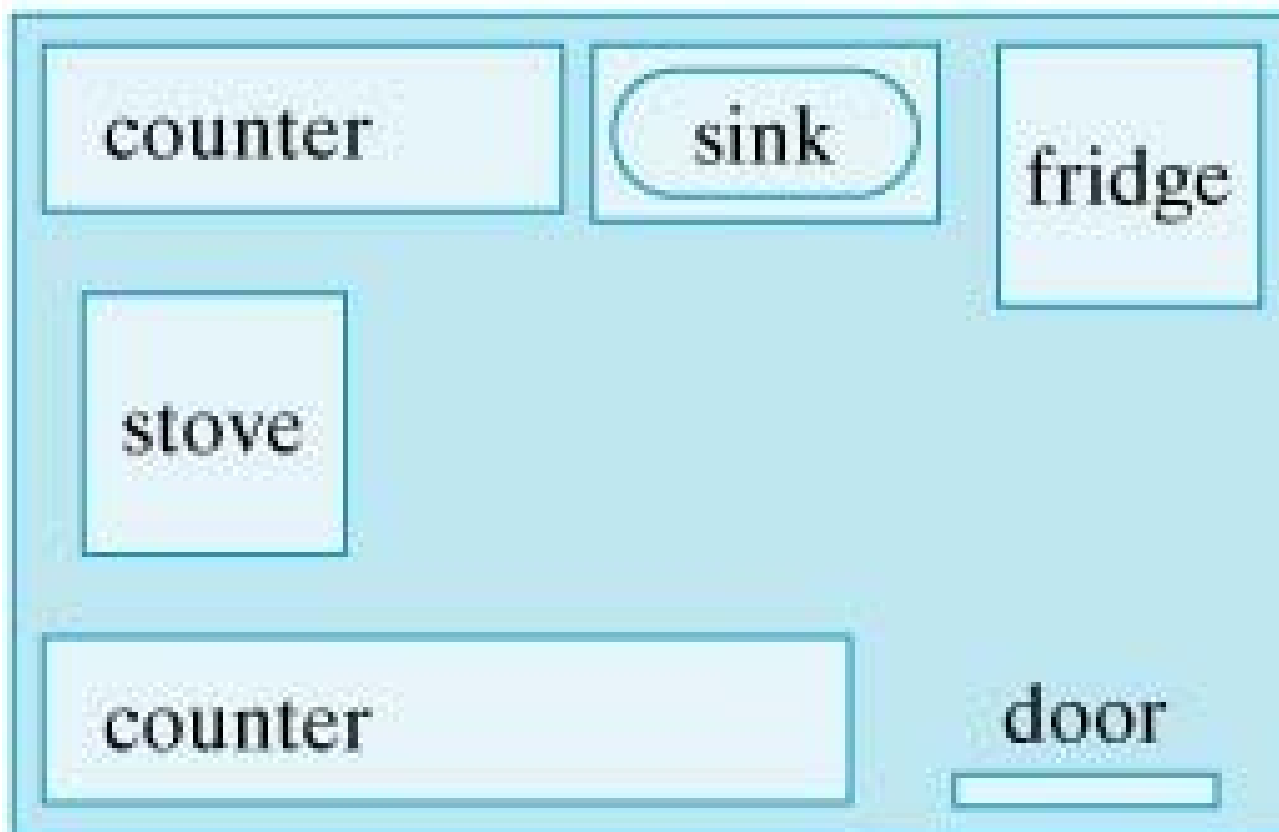
Литература:

Java How to Program, Sixth Edition, Chapter 1&2

2.1 Взаимодействия между обекти

- **Java е обектно ориентиран език**
 - **Програмата е съвкупност от обекти, които си взаимодействат**
 - **Telephones, houses, traffic lights, microwave ovens**
 - **Компютърната програма като съвкупност от взаимодействащи си обекти**

2.1 Взаимодействия между обекти



Модел на кухня

2.1 Взаимодействия между обекти

- **Видове обекти**

- **Активни (извършват действия самостоятелно)- шофьор, притежател на банкова сметка, компютерна програма**
- **Пасивни (нямат собствено поведение, други обекти предизвикват техни действия) – кола, банкова сметка, модел на кухня**
- **Обектите в компютърната програма си взаимодействат**
- **Шофьорът стартира двигателя на колата**
- **Притежател на банкова сметка тегли пари от АТМ**

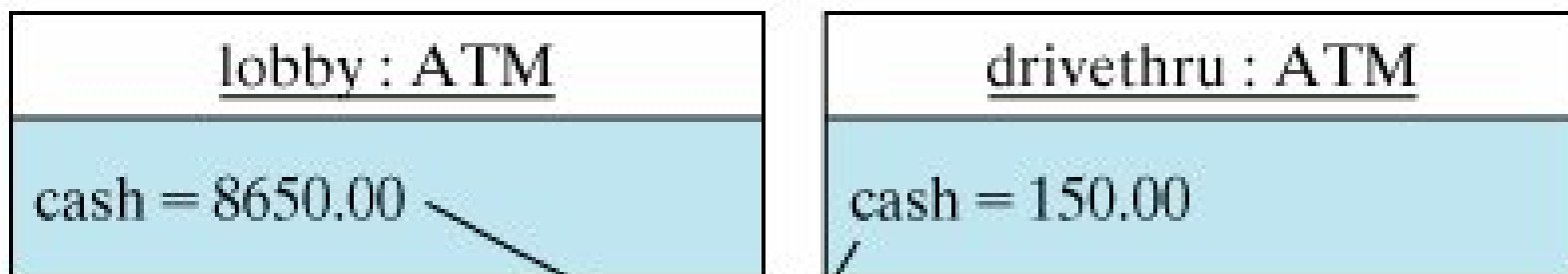
2.2 Представяне на обекти в ООП

- **Групи от обекти с общи характеристики**
 - Атрибути (текущ статус) *size, shape, color and weight*
 - Поведение *ball rolls, bounces, inflates and deflates; a baby cries, sleeps, crawls, walks and blinks; a car accelerates, brakes and turns - deposit(), withdraw(), viewBalance()*
- **Обект- модел на реални неща или събития**

2.3 Атрибути и стойности

- Изучаваме обектите като изследваме какво характеризира текущия им статус (атрибутите) и поведение (действията, които могат да извършват)
 - Cars / Trucks and Rectangles / Triangles
 - ATM атрибути
 - location , cash , dateOfLastCashReload
 - ChessPiece атрибути
 - *row* and *column* attributes
- Всеки атрибут се представя от определен тип данни

2.2 Представяне на обекти в ООП



An object's attributes and their values are shown in a second partition

Две АТМ и техните атрибути

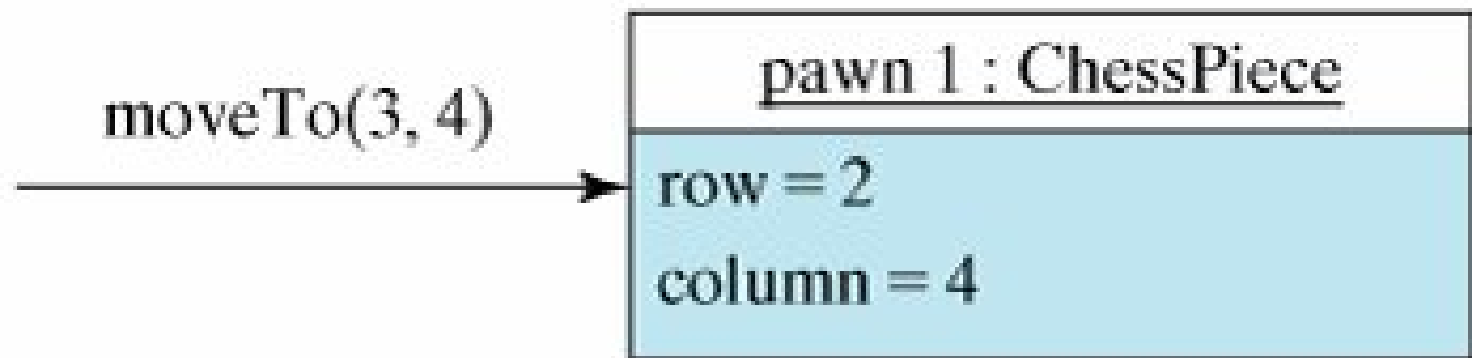
2.4 Действия и Съобщения

- Програмите имат активен характер-изпълняват действия, реализират поведение чрез изпълнение на серия от инструкции
 - *ChessPieces* премества на ново място по шахматната дъска
 - *moveTo()* a new position – извършва действие
 - АТМ- текущ баланс
 - АТМ to *report()* – връща стойност
- Действията винаги са свързани с определен обект, който ги реализира

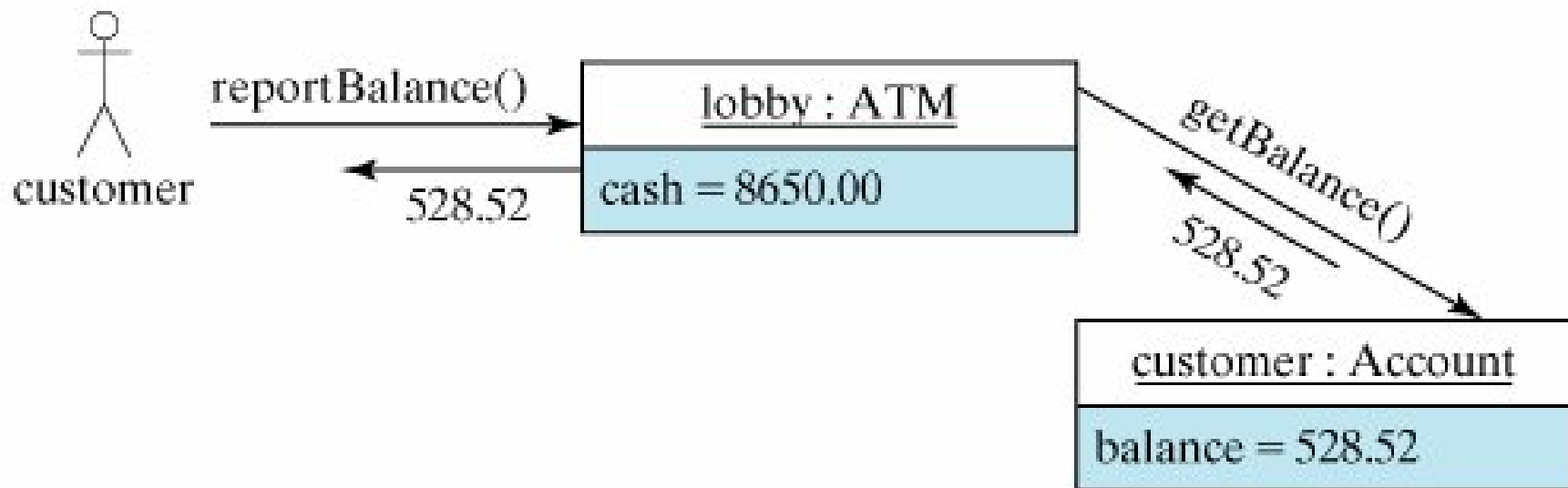
2.4 Действия и Съобщения

- **Съобщение-** предаване на информация или данни от един обект н адруг
 - *ChessPieces* премества на ново място по шахматната дъска
 - *moveTo (3,4)* – аргументи (3,4) указват къде да преместим пешката
- **Аргумент-** определя съдържанието на съобщението по определен начин
 - *moveTo (4,4)* – съобщение за преместване с 2 реда напред

2.4 Действия и Съобщения



2.4 Действия и Съобщения



2.4 Действия и Съобщения

- **За да се отговори на съобщение, обектът до когото е съобщението трябва знае как да изпълни желаното действие**
 - *АТМ трябва да знае как да пресметне баланса на потребителя*
 - **Как да се премества на ново място по шахматната дъска**
- **Обект може да отговаря на съобщения, които са в съответствие с неговите характеристики-статус и поведение**

2.4 Действия и Съобщения

- Видове съобщения

- Променят статуса на обекта до когото е съобщението- *mutator*

- moveTo()*

- Withdraw(500)*

- НЕ Променят статуса на обекта до когото е съобщението- *accessor*

- reportBalance()*

- currentCol()*

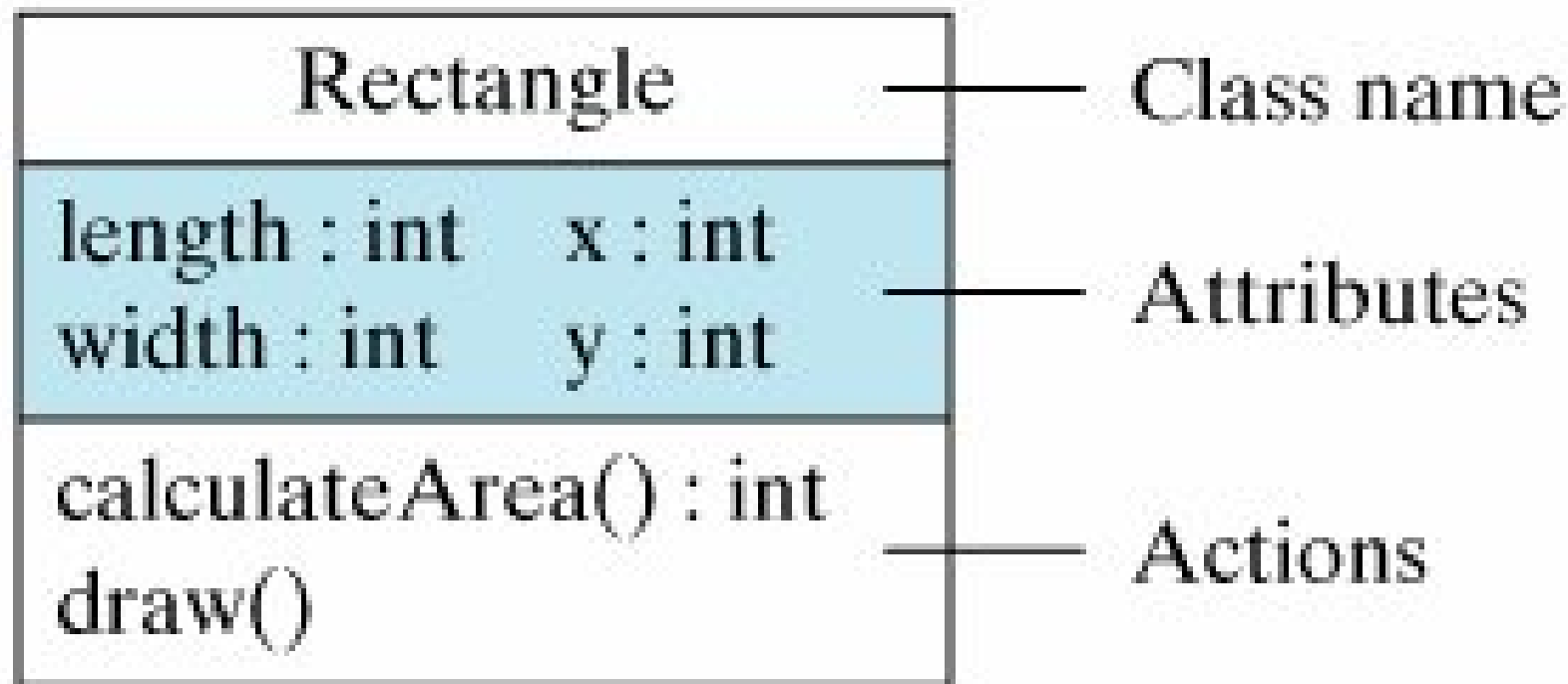
2.5 Класове в ООП

- **Всеки клас се характеризира с**
 - *Поле* на данните
 - *Множество от методи за операции с данните и предлагащи услуги на клиент класове- тези които ще изпращат съобщения*
- *Програмистите използват съществуващи класове като градивни елементи за конструиране на нови класове*
- *Всеки клас е като матрица за създаване на обекти с*
 - *еднакво описание на техния статус*
 - *Еднакво поведение- набор от методи за реализирането му*

2.5 Класове в ООП

- Всяка програма в ООП е клас само по себе си
- **Класът е програмата записана на файл**
- Всяко стартиране на програмата, създава обект от този клас, който се изпълнява като процес в оперативната памет на компютера
- Поведението на програмата характеризиращ нейното поведение се описва на Java от метода **main()**
- **Друг пример – програма за рисуване на правоъгълник**
- **Rectangle(length, width)**

2.5 Класове в ООП



2.6 Променливи и методи в ООП

- Действия и атрибути характеризират даден обект
- В термините на програмирането това се свежда до променливи и методи
- Променлива- съответства на атрибут
Именувано място в паметта за съхраняване на определен тип стойност
- Метод съответства на действие
- *Набор от програмни инструкции, които реализират определено действие в съответствие със статуса на обекта `rect.calculateArea()`*

2.6 Променливи и методи на обект и клас

- Променливи и методи могат да са определящи **само за даден обект или за всички обекти** от даден клас

Променлива или метод, принадлежащи на даден обект- *инстанция* на променлива или *инстанция* метод

Променлива или метод, **обща за всички обекти** от даден клас- клас променлива или клас метод

В Java клас променлива или клас метод се означава с ключовата дума *static*

2.6 Променливи и методи на обект и клас

- Пример

class Rectangle може да има

- *променлива nRectangles* , която съдържа текущия брой инстанции (обекти) от **class Rectangle** (*nRectangles е обща за всички обекти на class Rectangle*)
- *метод - main ()* общ за всички обекти на **class Rectangle** и да служи например, за създаване на нови обекти на **class Rectangle**

2.6 Променливи и методи на обект и клас

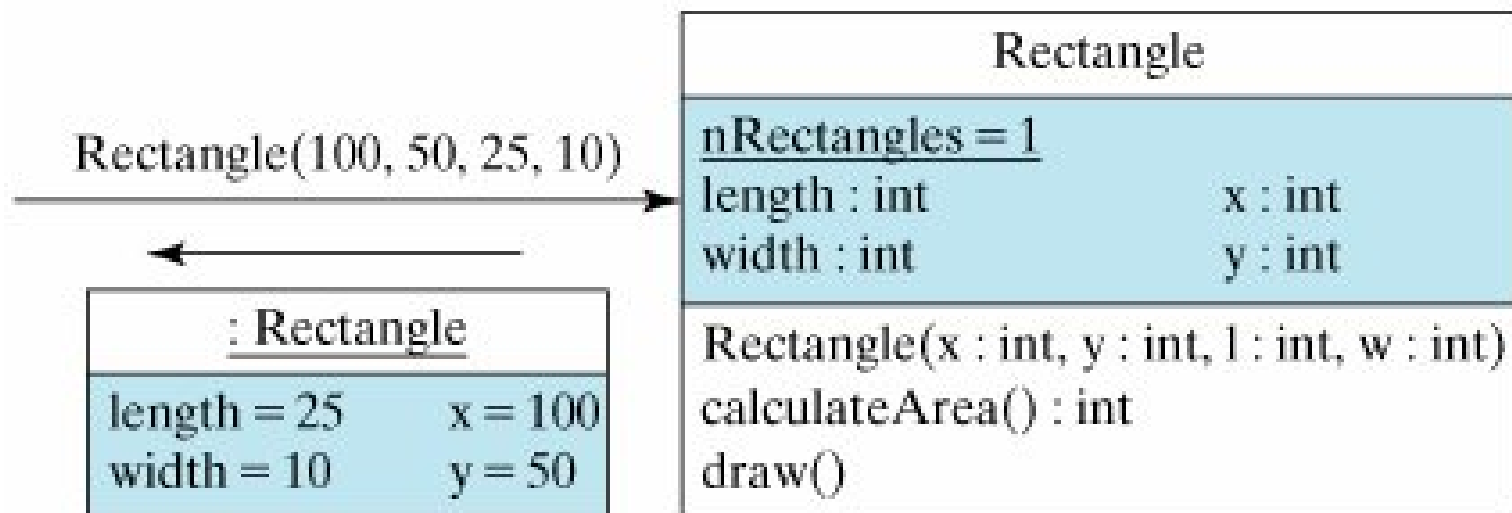
class променливата , *nRectangles*

Rectangle	
<u>nRectangles = 2</u>	
length : int	x : int
width : int	y : int
Rectangle(x : int, y : int, l : int, w : int)	
calculateArea() : int	
draw()	

<u>rect1 : Rectangle</u>	
length = 25	x = 100
width = 10	y = 50

<u>rect2 : Rectangle</u>	
length = 30	x = 125
width = 20	y = 75

2.6 Променливи и методи на обект и клас



Създаване на инстанция от class `Rectangle`

2.7 Първа Програма на Java

- **Java приложно програмиране**
 - **Обектно ориентиран език**
 - **Кратка програма за илюстрация на**
 - **Използването и структурата на Java класове**
 - **Първоначално запознаване със Java синтаксиса**
 - **Логиката в изпълнение на Java програмата**

2.7 Първа Програма на Java

- **Java приложение**

- **Необходим** е клас (**активен**), който да **стартира** изпълнението на програмата
- При изпълнението неговия **main()** метод **се създава обект от този клас**, който **от своя страна създава останалите обекти от класовете(пасивни)**, участващи в приложението
- Изпълнението на **програмата започва със зареждането на обект от програмния клас** в паметта и интерпретацията му от **Java Virtual Machine (JVM)**, чрез изпълнение на командата **java** от командния ред

- **Пример**

- Отпечатване на ред от текст
Welcome to Java Programming!
- Илюстрация на основни програмни единици

Outline

welcome1.java

```
1 // Fig. 2.1: welcome1.java
2 // Text-printing program.
3
4 public class welcome1
5 {
6     // main method begins execution of Java application
7     public static void main( String args[] )
8     {
9         System.out.println( "Welcome to Java Programming!" );
10
11     } // end method main
12
13 } // end class welcome1
```

Welcome to Java Programming!



2.7 Първа Програма на Java(прод.)

```
1 // Fig. 2.1: welcome1.java
```

- Коментари в края на реда- започват с : //
- Игнорират се при изпълнението на програмата
- Документират и описват кода
- Допринасят за читаемост на кода
- Изискване на професионално програмиране
- Обичайни, многоредови коментари: /* ... */
- /* This is a traditional
comment. It can be
split over many lines */

```
2 // Text-printing program.
```

Това е един ред коментар

- Заб. : номерацията редовете на програмата не са част от програмния код

Обичайна грешка при програмиране 2.1

Забравянето на някой от **ограничителите** на коментарите е синтактична грешка. **Синтаксисът** на програмния език определя правилата за писане на програма на съответния език. **Синтактична грешка** възниква, когато компилаторът открие код, който нарушава синтаксиса на Java В такъв случай, компилаторът не създава съответния `.class` файл. Вместо това се издава съобщение за грешка, която подпомага намирането и отстраняването на грешката в кода. Синтактичните грешки се наричат също грешки при компилация *compiler errors, compile-time errors or compilation errors*, защото се откриват във фазата на компилация. Не е възможно да се изпълни програмата, докато не се отстранят синтактичните грешки.

Правила за добро програмиране 2.1

Всяка програма да започва с коментар, обясняващ целта на програмата, описващ автора, датата и часа на нейното последно обновяване.

2.7 Първа Програма на Java(прод.)

3

- Празен ред
 - Прави програмата читаема
 - Празни редове, шпации(празни позиции) и табулация се игнорират от компилатора

4 `public class` welcome1

- Започва декларацията на програмния клас за `class Welcome1`
 - Всяко приложение на Java има поне един потребителски дефиниран клас
 - Ключови думи: думи запазени за употреба само за езика Java
 - `class` е ключова дума следвана от `class` име
 - Именуване на класове: **Всеки клас да започва с главна буква**
 - `SampleClassName`
 - Стил на “камила”- редуване на големи и малки букви
 - Всяка съставна дума започва с главна буква
 - Включва само символи позволени за идентификатори на Java

Правила за добро програмиране 2.2

Използвайте празни редове и празни символи за подобряване на читаемостта н апрограмата

2.7 Първа Програма на Java(прод.)

```
4 public class welcome1
```

– Java идентификатори

- Серия от символи състояща се от букви, цифри подчертаване (`_`) и символ (`$`)
- Не може да се започва с цифра
- Не може да съдържа празни символи(шпации)

Примери: `welcome1`, `$value`, `_value`, `button7`

- `7button` е невалиден идентификатор
- Java е зависима от големи и малки букви(capitalization matters)
 - `a1` и `A1` са различни идентификатори
- Използваме(засега) само `public class` (да не се спестява при писане на програмите)

2.7 Първа Програма на Java(прод.)

```
4 public class welcome1
```

– Java **КЛЮЧОВИ** думи

- Запазени от Java , имат **специалено предназначение**
- **Не може** да има идентификатори с иена на ключови думи
- Пишат се **само с малки** букви

Примери: `class`, `public static`, `int`, `void`

- Спазва се определен порядък когато участват в групи
 - `public static void` -> **правилно**
 - `void public static` -> **грешно**
 - `public class` -> **правилно**
 - `class public` -> **грешно**

2.7 Първа Програма на Java(прод.)

Table 1. Java keywords.

<code>abstract</code>	<code>continue</code>	<code>for</code>	<code>new</code>	<code>switch</code>
<code>assert</code>	<code>default</code>	<code>goto</code>	<code>package</code>	<code>synchronized</code>
<code>boolean</code>	<code>do</code>	<code>if</code>	<code>private</code>	<code>this</code>
<code>break</code>	<code>double</code>	<code>implements</code>	<code>protected</code>	<code>throw</code>
<code>byte</code>	<code>else</code>	<code>import</code>	<code>public</code>	<code>throws</code>
<code>case</code>	<code>enum</code>	<code>instanceof</code>	<code>return</code>	<code>transient</code>
<code>catch</code>	<code>extend</code>	<code>int</code>	<code>short</code>	<code>try</code>
<code>char</code>	<code>final</code>	<code>interface</code>	<code>static</code>	<code>void</code>
<code>class</code>	<code>finally</code>	<code>long</code>	<code>strictfp</code>	<code>volatile</code>
<code>const</code>	<code>float</code>	<code>native</code>	<code>super</code>	<code>while</code>

Правила за добро програмиране 2.3

По приетия стандарт за стил **ВИНАГИ** започвайте името на клас с **Главна буква**, а също и всяка следваща дума, участваща в името на класа- **останалите букви да са малки**. Java професионалните програмисти именуват класовете по такъв начин- **стил на “камила”**.

Същото, естествено важи и за имената на файловете, съдържащи *public class* на Java

Пример: *MyFirstClass*, *BankAccount*, *ChessBoard*

MyFirstClass.java, *BankAccount.java*, *ChessBoard.java*

Правила за добро програмиране 2.3

По приетия стандарт за стил **ВИНАГИ** **започвайте** името методи и променливи с **малка буква**, а всяка **следваща дума**, участваща в методи и променливи да започва с **Главна буква**- **останалите букви** да са малки. Java професионалните програмисти именуват класовете по такъв начин- **стил на “камила”**

Пример:

```
main( ), computeTotal( ),  
accountBalance, countItems, xCoord, yCoord
```

Обичайна грешка при програмиране 2.2

Java е чувствителна по отношение на използването на големи и малки букви. Използването на погрешна последователност от големи и малки букви при рефериране към един и същ идентификатор води до синтактични грешки.

2.7 Първа Програма на Java(прод.)

```
4 public class welcome1
```

– Запазване на файла с Java кода

- **Името на файла** трябва да е същото както името на **public** класа и да има **.java** окончание
- **Welcome1.java**

```
5 {
```

– Лява фигурна скоба {

- Започва тяло на код- в този случай тялото на **class Welcome1**

– Дясна фигурна скоба } затваря декларацията на класа (ред 13)

Обичайна грешка при програмиране 2.3

Грешка е за `public class` да има име за файл , което е различно от името на този клас (plus `.java` окончанието) по отношение на ред от символи и последователност от големи и малки букви.

Обичайна грешка при програмиране 2.3

Грешка е файл с окончание `.java` да съдържа повече от един `public class` на Java.

Пишете всеки `public class` на Java в **отделен файл!**

Обичайна грешка при програмиране 2.4

Грешка е да се пропусне окончанието .java
за файл, съдържащ Java програмен код.
При отсъствие на окончанието
компиляторът няма да може да компилира
класа.

Правила за добро програмиране 2.4

Когато пишете отворена скоба, {, в програмата си, веднага пишете и затваряща скоба. Така избягвате излишни синтактични грешки.

Правила за добро програмиране 2.5

Подравнявайте кода на клас декларациите между, {, и, } както е дадено в примерния код. Това допринася за читаемост и по-лесно поправяне на синтактични грешки.

Обичайна грешка при програмиране 2.5

Липсата на някоя от отваряща или затваряща скоба е синтактична грешка.

2.7 Първа Програма на Java(прод.)

```
7 public static void main( String args[] )
```

- Всяко Java приложение трябва да има програмен клас с ***main()*** метод дефиниран както е показано
 - Изпълнението на програмата започва от метода ***main()***
 - Кръглите скобки в ***main*** указват, че **това е метод**
 - ***static*** означава, че това е **клас метод** (*общ за всички обекти*)
 - Java приложенията **съдържат един или повече методи**
 - Точно един клас метод може да се казва ***main***
- Методите могат да изпълняват задания и да връщат информация
 - ***void*** означава, че ***main*** метода **не връща** информация
 - Методът ***main(String[] args)*** взима аргумент ***String[] args***
 - Използвайте **винаги** ***main(String[] args)*** с аргументи

Лява скоба започва начало на тялото на метода ***main()***

```
8 {
```

- Дясна скоба завършва тялото ***main()*** на метода ***}*** (line 11)

Правила за добро програмиране 2.7

Подравнявайте цялото тяло на всеки метод на едно ниво, както е показано в примерния код.

2.7 Първа Програма на Java(прод.)

```
9      System.out.println( "Welcome to Java Programming!" );
```

- Инструкция до **computer** да изпълни действие
 - Отпечатва низ от символи
 - **String** – низ от символи в двойни кавички
 - Празните символи в низа НЕ СЕ игнорират от компилатора
- **System.out**
 - Обект използван за **стандартен изход**
 - Отпечатва в **command window** (i.e., MS-DOS prompt)
- Използва метода **System.out.println**
 - Отпечатва низ от символи
 - Взима **един единствен аргумент** от тип **String**
- Това е пример за изпълнима инструкция
 - Всяка изпълнима инструкция завършва с **;**
 - Инструкцията принадлежи към обекта **System.out**
 - Инструкцията и обекта ѝ са разделени с точка

Обичайна грешка при програмиране 2.6

Пропускането на *точка и запетая* в края на инструкцията е синтактична грешка. Опитайте, какво ще се случи при компилация, ако пропуснете да пишете ; в края на *изпълнима инструкция*.

Поправка на грешки Tip 2.2

При грешка от компилация, получавате номера на реда със синтактична грешка . Ако този ред няма грешка, проверете на предхождащите редове за синтактична грешка.

2.7 Първа Програма на Java(прод.)

```
11      } // end method main
```

- **Край на декларацията на метода**

```
13 } // end class welcome1
```

- **Край на декларацията на класа**
- **Може да се добавят коментари за проследяване края на затварящите скоби**

Правила за добро програмиране 2.8

Използването коментари при (}) в края на тяло на клас или метод допринася за по-добро четене и трасиране за грешки на програмата.

2.7 Първа Програма на Java(прод.)

- **Compiling на програмата**

- Отворете Command prompt window, идете в директорията на `Welcome1.java`
- изпълнете `javac welcome1.java`
- Ако няма синтактични грешки се създава `welcome1.class`
 - съдържа **bytecode** -а който позволява изпълнението на програмата
 - **Bytecode** -а се **интерпретира** от JVM (виртуалната машина на Java)

Поправка на грешки Tip 2.3

Ако при компилацията се получи съобщение от рода на “bad command or filename,” “javac: command not found” или “'javac' is not recognized as an internal or external command, operable program or batch file,” то Java не е инсталирана правилно.

По специално *PATH* environment променлива не е зададена правилно.

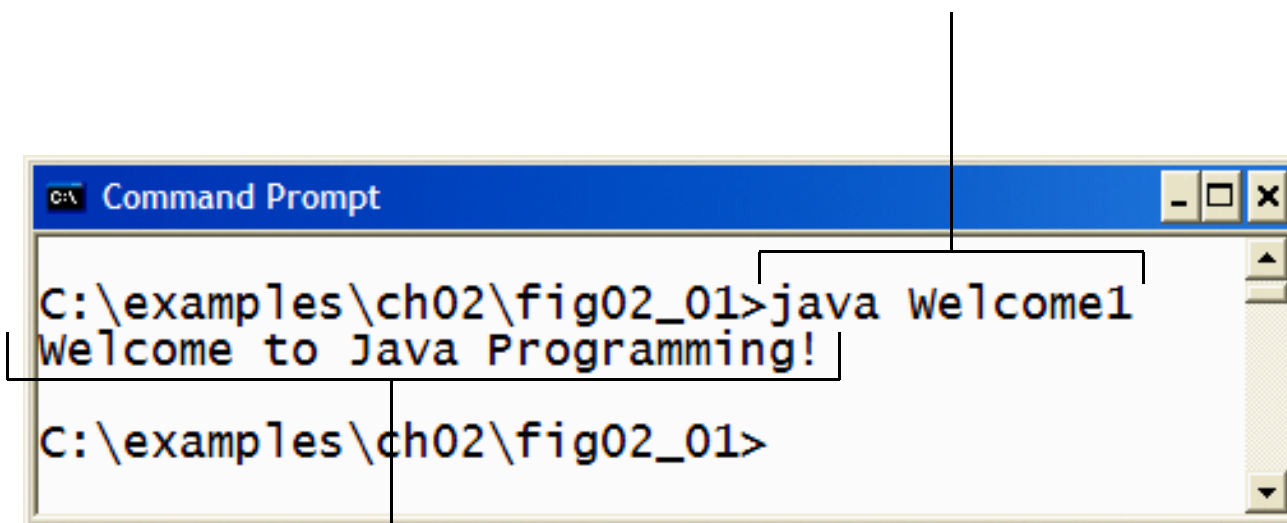
Поправка на грешки Tip 2.5

Ако компилаторът даде съобщение за грешка “`public class ClassName must be defined in a file called ClassName.java`” то името на файла не съвпада с името на `public class` в този файл или това файл име е неправилно написано при стартиране на компилатора.

2.8 Изпълнение на Програмата (прод.)

- **Изпълнение на програмата**
 - Изпълнението от командния ред `java Welcome1`
 - стартира JVM
 - JVM зарежда `.class` файла за клас `Welcome1`
 - `.class` окончанието може да се пропусне
 - JVM изпълнява метода `main()`

You type this command to execute the application



The image shows a screenshot of a Windows XP Command Prompt window. The title bar is blue and contains the text "C:\ Command Prompt" and standard window control buttons (minimize, maximize, close). The main area of the window is white and contains the following text: "C:\examples\ch02\fig02_01>java welcome1", "Welcome to Java Programming!", and "C:\examples\ch02\fig02_01>". A vertical line from the text "You type this command to execute the application" points to the command "java welcome1". A bracket from the text "The program outputs" points to the output "Welcome to Java Programming!".

```
C:\examples\ch02\fig02_01>java welcome1
Welcome to Java Programming!
C:\examples\ch02\fig02_01>
```

The program outputs

Welcome to Java Programming!

Fig. 2.2 | Изпълнение на welcome1 в Microsoft Windows XP Command Prompt window.

Поправка на грешки Tip 2.6

Ако при изпълнение на Java program, получите съобщение за грешка от сорта “Exception in thread “main” java.lang.NoClassDefFoundError: welcome1,” то *CLASSPATH* environment variable не е зададена правилно.

2.8 Обобщение

- **ОО е естествен начин на мислене за света и писането на компютърни програми.**
- **UML е графичен език, позволяващ ОО модели да се представят по един стандартен метод за използване на означения**
- **ООД (обектно ориентиран дизайн) моделира софтуерни компоненти в термините на реални обекти. ООД използва предимствата на взаимовръзките между класовете, където обектите от даден клас имат еднакви характеристики- статус и поведение. Използва се също предимството от релации на наследство, като новосъздадените класове придобиват по наследство характеристиките на базовите класове от които произлизат, а също и добавят специфични за тях нови характеристики.**
- **ООД капсулира(затваря) (атрибути) и функции (поведение) в обекти като данните и функциите на обект са свързани в едно цяло.**
- **Обектите притежават свойството на скриване на информация.- обектите в общия случай не знаят структурата на други обекти или как другите обекти се изпълняват.**

2.9 Обобщение

- Обектно ориентираното програмиране (ООП) позволява на програмистите да прилагат ООД в работещи компютърни системи.
- В Java, единицата на програмиране е **class** от който обектите евентуално се създават. Java програмистите се концентрират върху създаване на собствени класове и повторно използване на съществуващи класове. Всеки **class** съдържа **данни** и **функции** които манипулират тези данни. Компонентите функция се наричат **методи**..
- Една инстанция от **class** се нарича **обект**.
- Класовете влизат в релации с други класове. Тези релации могат да бъдат ИМА, ЗНАЕ-ЗА и Наследственост. Релациите ИМА и ЗНАЕ-ЗА се наричат общо като асоциации(associations).
- Посредством ОО технология, програмистите изграждат софтуер, който в по- голяма част използва стандартни библиотеки от класове.
- Процесът на анализ и моделиране на система от гледна точка на ОО се нарича обектно ориентиран анализ и дизайн (OOAD)

1.10 Задачи

1. **Анализирайте една лека кола в термините на ООП. Опишете съставлящите я компоненти като използвате класове. Какви са релациите на тези класове по отношение на класа, представящ леката кола?**
2. **Опишете взаимодействието между студент, библиотекар и базата данни на библиотеката, когато студента прави заявка за ададена книга от библиотеката. Използвайте за целта ознаения аналогични на Slide 13.**
3. **Анализирайте в термините на ООАД структурата на клас симулиращ работата на:**
 - Телефон
 - Асансьор
 - Компютър
 - Калкулатор
4. **Напишете програма на Java, която посредством символа ‘*’ отпечатва триъгълник, правоъгълник в текстовия прозорец на Command prompt-a**

1.10 Задачи

5. Кои от следните идентификатори са валидни ?

<i>int</i>	<i>74ElmStreet</i>	<i>Big_N</i>	<i>L\$&%#</i>
<i>Boolean</i>	<i>_number</i>	<i>big numb</i>	<i>public</i>
<i>Private</i>	<i>Joe</i>	<i>j1</i>	
<i>Int</i>	<i>2*K</i>	<i>boolean</i>	

6. Обяснете разликата между *клас променлива* и *променлива на инстанция(обект)*.

7. Каква е ролята на:

- Метода *main()*
- Конструктор метода