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 3.2

Въведение в програмирането с класове и обекти

- **Деклариране на клас и създаване на обекти**
- **Деклариране на данни на клас с оглед дефиниране на статуса на обектите в класа**
- **Деклариране на методи на клас с оглед дефиниране на поведението на обектите на класа**
- **Изпълнение на метод на обект**
- **Променливи клас данни и променливи данни на метод на обект**
- **Използване на конструктор за инициализиране на данните на обект при създаването му**
- **Задачи**

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

office@progressbg.net www.progressbg.net

Въведение :

Класове

- При програмиране на Java, основното е да се напишат дефинициите за класовете на отделните обекти, които ще изграждат програмата.

- Всяка дефиниция на клас капсулира данните на своите обекти и тяхното поведение.

- След като класът веднъж е дефиниран, той служи като шаблон за създаване на отделни обекти(инстанции) от класа

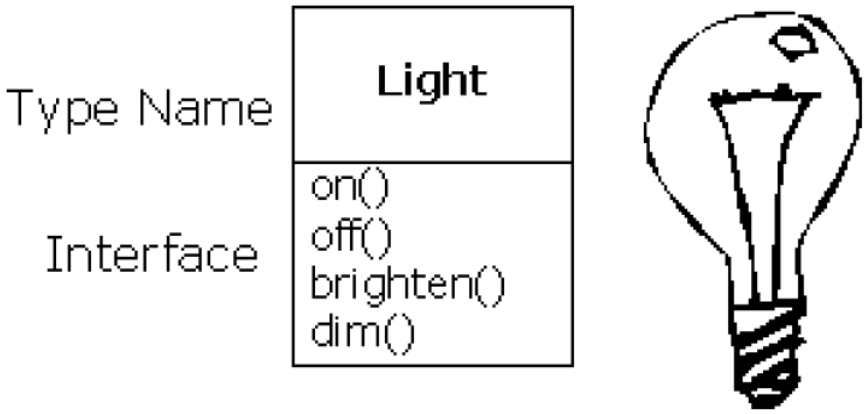
- Всеки клас съдържа две основни групи от елементи :

  - Променливи – служат за съхранение на данни, които определят текущото състояние на обектите по отделно или за всички обекти

  - Методи – служат за реализация на определено поведение(действие) или на всички обекти като цяло

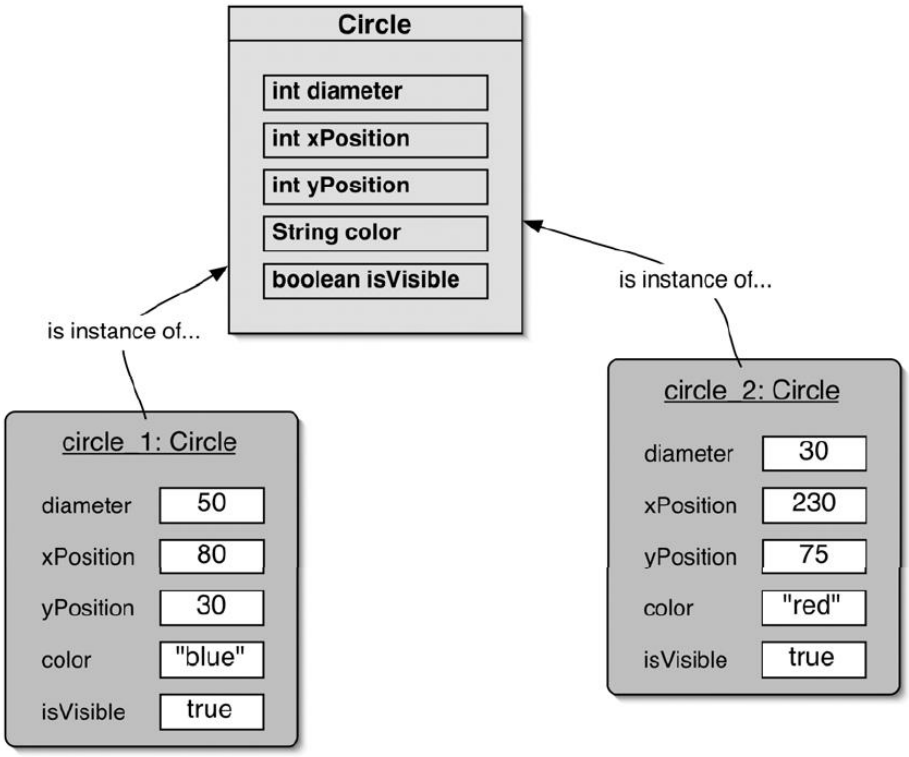
- За моделиране на класа, от който произхожда даден обект трябва да се намери отговор на следните въпроси:
  - Каква роля изпълнява обекта в програмата
  - Какви характеристики(данни) определят текущото състояние на обекта?
  - Какво поведение трябва да реализира обектът съобразно данните, описващи текущото му състояние?
  - Дали си комуникира с други обекти и по какъв начин?
  - Каква част от поведението на обекта трябва да се скрие и да е недостъпно за други обекти?

class Light – **методи** за клас осветление :



- Какви данни може да съдържа класът Light?

class Circle – данни за клас Circle



Всеки клас се дефинира, чрез сорс код на Java и описва елементите на класа(променливи и методи).

Всеки обект съдържа едно или повече полета дани

- Наричат се instance variables

- Те принадлежат на всеки обект от момента на създаването му и могат да се променят единствено от методи на същия този обект

Следват примери, допълващи последователно class GradeBook с нови елементи за демонстиране основни концепции в дефинирането на класове с Java

- **Първи пример**, дефинираме class GradeBook с един метод, който извежда **welcome** съобщение при изпълнението му.После демонстрираме как се създава обект от този клас и изпълняваме този метод,за да изведе **welcome** съобщение

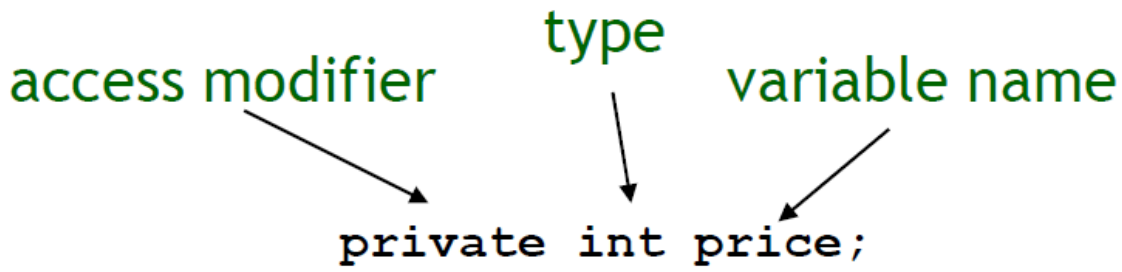
- **Втори пример,** допълва първия, позволявайки методът да получи за аргумент име-на-курс и извеждане на това име като част от welcome съобщението

- **Трети пример,** показва как да включим име-на-курс и да получим текущата стойност на име-на-курс

- **Четвърти пример,** демонстрира как да инициализираме данна(променлива,поле) на **GradeBook** object при създаване на обекта посредством конструктора на class GradeBook

Декларацията на всеки клас започва с ключовите думи public class и трябва да се запише във файл със същото име, под което е дефиниран класа, следвано от окончанието .java за име на файл.

```
public class Car
{
    private int    price;
    private float  speed;


                                    ...
}
```



- Разглеждаме програма, състояща се от два класа GradeBook (**пасивен клас**) и GradeBookTest(**активен клас**)
- class **GradeBookTest** ще стартира изпълнението, ще създаде от class **GradeBook** и ще извика метод от този клас за изпълнение.

- Деклариран в отделен файл с име GradeBook.java
- Моделира обект Дневник на курс (С какви данни и методи бихте моделирали класа Дневник на курс? Защо това е пасивен клас?)
- Трябва да е общо достъпен → ключова дума public се използва като модификатор на достъп(access modifier)
- **Декларацията на всеки клас включва**:
  - Модификатор на достъп
  - Ключовата дума class
  - Двойка скоби – лява и дясна фигурна скоба

GradeBook.java

```
1  // Fig. 3.1: GradeBook.java
2  // Class declaration with one method.
3
4  public class GradeBook
5  {
6     // display a welcome message to the GradeBook us
7     public void displayMessage()
8     {
9        System.out.println( "Welcome to the Grade Book!" );
10    } // end method displayMessage
11
12 } // end class GradeBook
```

GradeBook.java

Извежда ред от текст на Стандартен изход

Методът **displayMessage()** на class GradeBook

- Използва ключовата дума public за да укаже, че този **метод е общо достъпен**
- Използва ключовата дума void за да укаже, че този метод **не връща данни като резултат**

-Заглавието на този метод се нарича съвкупността от модификатора за достъп,типът на връщаните данни,името на метода и двойката фигурни скоби(лява и дясна), дефиниращи тялото на метода.

class GradeBookTest

Служи за стартиране на Java приложението

- Съдържа public static void main () метод

- Създава обектите на програмата и стартира взаимодействието по между им

- class GradeBookTest е пример за активен клас

- Всяко Java приложение трябва да има поне един такъв клас

Израз за **създаване** **на обект** от клас

- Използва се ключовата дума new

- Следвано от името на класа и кръглите скоби

Извикване на **метод на обект**

- Използва се означението имеНаОбект.имеНаМетод()

- Завършва с точка и запетая

- **Обектът трябва да е създаден,за да се извиква метод от този обект!**

GradeBookTest.java

```
1  // Fig. 3.2: GradeBookTest.java
2  // Create a GradeBook object and call its displayMessage method.
3
4  public class GradeBookTest
5  {
6     // main method begins program execution
7     public static void main( String args[] )
8     {
9        // create a GradeBook object and assign it to myGradeBook
10       GradeBook myGradeBook = new GradeBook();
11
12       // call myGradeBook's displayMessage method
13       myGradeBook.displayMessage();
14    } // end main
15
16 } // end class GradeBookTest
```

Създава обект от class GradeBook

Извиква метод displayMessage на обект GradeBook

```
Welcome to the Grade Book!
```

UML Class диаграма за class GradeBook

UML клас диаграми – правоъгълници, разделени вертикално на три части :

- Горната част съдържа името на класа

- Средната част съдържа данните на класа или т.нар. променливи

- Долната част съдържа описание на имената и типа данни за всеки метод на класа

- **Плюс знак** означава public метод или данна
- **Минус знак** означава private метод или данна

UML диаграма на class GradeBook

Деклариране на методи, използващи аргументи

Аргументи на методи :

- Предават допълнителна информация на метод(**форматирани аргументи**) – част от заглавието на метода

- Подават се при изпълнението на метод, използващ аргументи(**реални аргументи**) – реалните стойности, предвани на метода при изпълененението му

GradeBook.java

```
1  // Fig. 3.4: GradeBook.java
2  // Class declaration with a method that has a parameter.
3
4  public class GradeBook
5  {
6     // display a welcome message to the GradeBook user
7     public void displayMessage( String courseName )
8     {
9        System.out.printf( "Welcome to the grade book for\n%s!\n",
10          courseName );
11    } // end method displayMessage
12
13 } // end class GradeBook
```

Формален аргумент за ***displayMessage()***

Изпълнение на метод ***printf()*** като courseName в случая е реален аргумент за ***printf()***

Методи на обекти от class Scanner :

- nextLine() Прочита следващия ред от Стандартен вход като String
- next() Прочита следващата дума(разделител е празен символ) от Стандартен вход като String
- nextInt() прочита следващата дума от Стандартен вход като цяло число
- nextDouble() прочита следващата дума от Стандартен вход като число с плаваща запетая с двойна точност

GradeBookTest.java

```
1  // Fig. 3.5: GradeBookTest.java
2  // Create GradeBook object and pass a String to
3  // its displayMessage method.
4  import java.util.Scanner; // program uses Scanner
5
6  public class GradeBookTest
7  {
8     // main method begins program execution
9     public static void main( String args[] )
10    {
11       // create Scanner to obtain input from command window
12       Scanner input = new Scanner( System.in );
13
14       // create a GradeBook object and assign it to myG
15       GradeBook myGradeBook = new GradeBook();
16
17       // prompt for and input course name
18       System.out.println( "Please enter the course name
19       String nameOfCourse = input.nextLine(); // read a
20       System.out.println(); // outputs a blank line
21
22       // call myGradeBook's displayMessage method
23       // and pass nameOfCourse as an argument
24       myGradeBook.displayMessage( nameOfCourse );
25    } // end main
26
27 } // end class GradeBookTest
```

Изпълнява метод nextLine да прочете целия въведен ред

Изпълнява метод displayMessage с един аргумент

Output :

Please enter course name:

→Introduction to Java programming with Progress

In this case the method prints:

„Welcome to grade book for

Introduction to Java programming with Progress!”

✓ Обикновено обекти се създават с ключовата дума **new.**Константите(наричани още литерали) от тип String се представят в двойни кавички като например “hello”.String константите референции към String обекти, които неявно се създават от компилатора на Java.

Обичайна грешка при програмиране :

- Грешка при компилация е, когато в тялото на метода има променлива със същото име,както някой от аргументите на метода в неговата дефиниция.

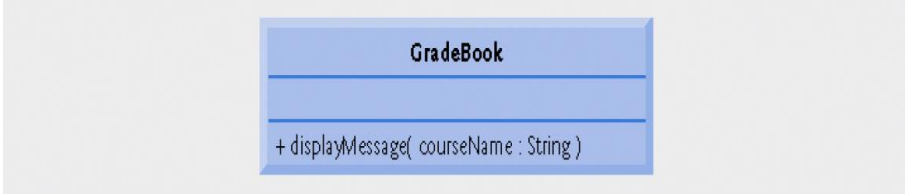
Списъкът с аргументи на един метод задва неговия ”подпис”, характеризира се с :
- Брой на аргументите
- Поредност на типовете на аргументите

Два метода са **различни**,ако имат
- Различно име
- Еднакво име,но различен подпис

Обичайна грешка при програмиране :

- Грешка при компилация се издава винаги, когато броят на реалните аргументи е различен от броя на формалните аргументи.Иначе казано,ако извикаме метод с различен брой аргументи от броя на дефинирани в декларацията.Също така реалните аргументи не съотвестват по тип и по ред на списъка от формални аргументи,с които е дефиниран метода.

UML class diagram за class GradeBook :



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

office@progressbg.net www.progressbg.net

Бележки върху import декларациите

- ✓ java.lang се импортира неявно във всеки клас на Java

- ✓ Пакет по подразбиране(default package)
  -Съдържа всички class файлове в текущата директория
  - Тя също се импортира неявно в source code на всеки Java файл, намиращ се в текущата директория

  Например,ако файловете Circle.java и DrawTest.java са в една и съща директория, то class DrawTest може да използва class Circle без да има нужда от import декларация за class Circle, понеже DrawTest.java се намира в същата директория, заедно с Circle.java.Същото важи и за class GradeBook и class GradeBookTest

- ✓ import замества необходимостта от използване на името на съответната библиотека пред името на класа – напълно определено име на клас

- ✓ Например,ако **не** използвате пред дефиницията на даден клас import java.util.Scanner; то вътре в дефиницията на клас трябва да пишете всеки път java.util.Scanner // , когато искате да използвате клас Scanner
- ✓ Java компилаторът **не** изисква import декларации в Java source файла, ако **навсякъде** се използва напълно определеното име на класа(пред името на класа стои библиотеката, към която принадлежи класа).

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

office@progressbg.net www.progressbg.net

Данни на клас, set методи и get методи

Локални променливи

- Декларирани в тялото на метода
- Включват аргументите на метода
- Достъпни са **единствено** в тялото на метода!

Данните,декларирани в тялото на класа и извън методите му

- Наричат се данни,променливи,атрибути на клас
- Достъпни са **във всеки един от методите** на класа
- Декларират се **наедно в началото на тялото на класа**(преди всички методи),а не разхвърляни между методите на класа
- Всеки обект от класа получава отделно копие(инстанция) от данните на класа (в случай, че съответната данна не е обща за всички обекти)

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

office@progressbg.net www.progressbg.net

Модификатори за достъп public и private

private ключова дума

- Означава,че данна или метод е **достъпна единствено в тялото на своя клас на дефиниция**

- Винаги дейкларирайте клас данните като private

- Реализира data hiding(скриване на информация) и гарантира контролиран достъп до данните и метода на класа.

public ключова дума

- Означава,че данна или метод, е общо достъпна за всички обекти и класове в изпълняваната програма
- В рамките на този курс **никога не** декларирайте клас данни **public**
- Винаги декларирайте **set** и **get** методите като public
- Класовете също трябва да са **public**

Return тип на метод

- Тип на данните, връщани като резултат от изпълнението на метода
  - Декларира се в заглавеието на метода
  - Използва командата return, следвана от променлива,константа или стойност от същия тип
  - return тип е void, когато методът няма предназначение да връща данни

```
public void insertMoney(int amount)
{
    balance = balance + amount;
}  // no return type
```

```
public int insertMoney(int amount)
{
    balance = balance + amount;
    return balance;
} // with return type
```

Set методи

Методи за промяна(mutator methods)

- Водят до промяна в стойността на клас данната
- Служат, за да задат нова текуща стойност на една или повече клас данни от други обекти, на които е позволено такова действие
- Задължително се проверява валидността на новата стойност
- Регулират кой и как може да променя данните на класа

Наричат се още **set методи**

- Задължително името съдържа префикс **set**, следвано от името на клас данна, за чиято стойност ще се променя с метода.Първата буква в името на клас данната се сменя с главна

- По правило клас данните трябва да са недостъпни(**private**) за други обекти

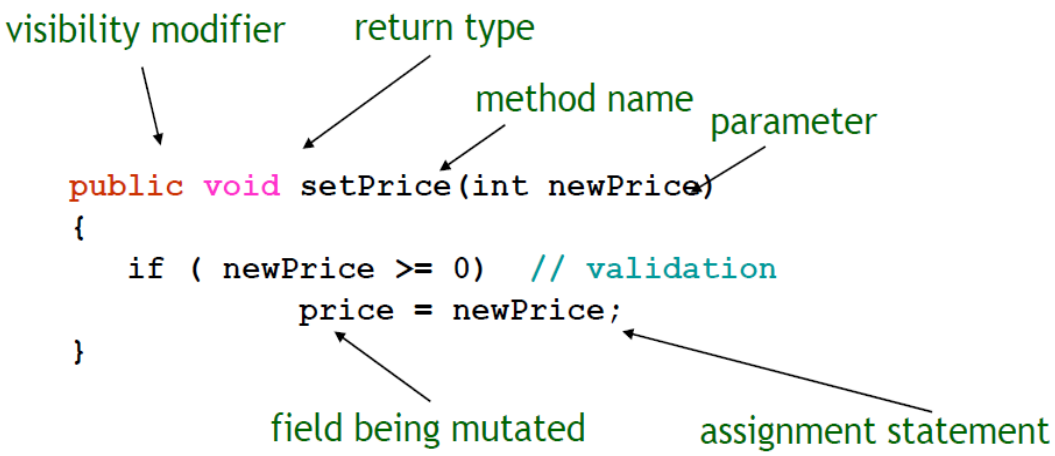
- Клас данна трябва да се променя единствено чрез съответен set метод

Пример :

```
private String courseName;        // клас данна
public void setCourseName(String newCourse){ // set метод за courseName
      if (newCourse != null)  // validate!
                  courseName = newCourse;
}
```

Set метод за цена :

```
private int price;
```



- Примери за валидация при примерни set методи?

Get методи

Методи за достъп(**accesor methods**)
- Не водят до промяна в стойността на клас данната

- Служат да дадат информация за текущата стойност на една или повече клас данни, за които другите обекти е позволено да узнаят

- Регулират достъпа до данните на класа

- Наричат се още **get методи**
  - Задължително името съдържа префикс **get**, следвано от **името на клас данна**, за чиято стойност ще се дава информация с метода, като първата буква в името на клас данната се сменя с главна за други обекти

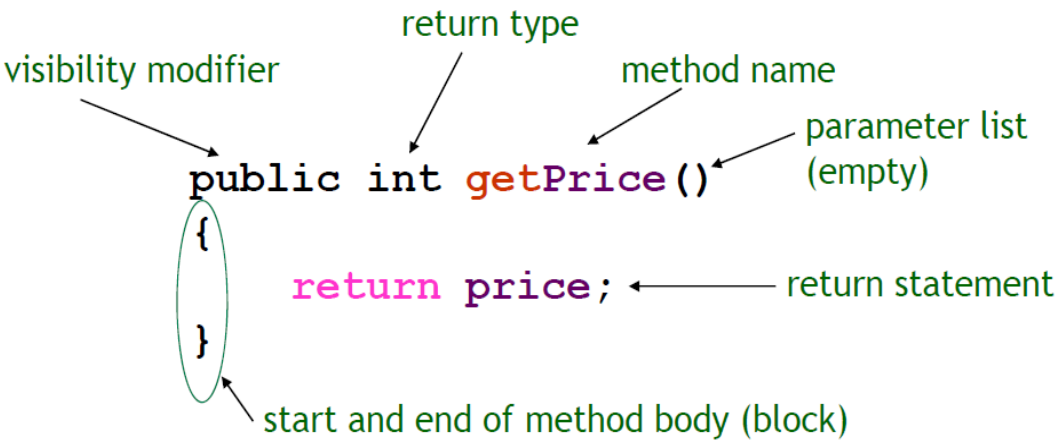
  - Клас данната се прави достъпна чрез дефиниране на съответен get метод

Пример :

```
private String courseName;          // клас  данна
public  String  getCourseName(){ // get метод  за courseName
          return courseName;
}
```

Пример

```
private int price;
```



Продължение на GradeBook

```
1  // Fig. 3.7: GradeBook.java
2  // GradeBook class that contains a courseName instance variable
3  // and methods to set and get its value.
4
5  public class GradeBook
6  {
7     private String courseName; // course name for this GradeBook
8
9     // method to set the course name
10    public void setCourseName( String name )
11    {
12       courseName = name; // store the course name
13    } // end method setCourseName
14
15    // method to retrieve the course name
16    public String getCourseName()
17    {
18       return courseName;
19    } // end method getCourseName
20
21    // display a welcome message to the GradeBook user
22    public void displayMessage()
23    {
24       // this statement calls getCourseName to get the
25       // name of the course this GradeBook represents
26       System.out.printf( "Welcome to the grade book for\n%s!\n",
27          getCourseName() );
28    } // end method displayMessage
29
30 } // end class GradeBook
```

Клас данна courseName

set метод за courseName

get метод за courseName

Извикване на *get* метод

Правила за добро програмиране :

- ✓ **ВИНАГИ** декларирайте клас данните в началото на класа преди методите.

- ✓ Оставяйте по един празен ред между дефинициите на методите за придаване на по-добра читаемост на програмите

Модификация на GradeBookTest

- Създава обект myGradeBook от class GradeBook
- Демонстрира поведението на така получения обект, чрез изпълнение на методите на този обект, достъпни за class GradeBookTest
  -Разпечатва текущата стойност на клас данна courseName на обекта myGradeBook чрез **get** Метода
  - Референтните данни са **null** по подразбиране
  - Прочита нова стойност за courseName от Стандартен вход и присвоява на courseName чрез **set** метода

```
1  // Fig. 3.8: GradeBookTest.java
2  // Create and manipulate a GradeBook object.
3  import java.util.Scanner; // program uses Scanner
4
5  public class GradeBookTest
6  {
7     // main method begins program execution
8     public static void main( String args[] )
9     {
10       // create Scanner to obtain input from command window
11       Scanner input = new Scanner( System.in );
12
13       // create a GradeBook object and assign it to myGradeBook
14       GradeBook myGradeBook = new GradeBook();
15
16       // display initial value of courseName
17       System.out.printf( "Initial course name is: %s\n\n",
18          myGradeBook.getCourseName() );
19
```

Изпълнява *get* метода за courseName

```
20       // prompt for and read course name
21       System.out.println( "Please enter the course name:" );
22       String theName = input.nextLine(); // read a line of text
23       myGradeBook.setCourseName( theName ); // set t
24       System.out.println(); // outputs a blank line
25
26       // display welcome message after specifying course name
27       myGradeBook.displayMessage();
28    } // end main
29
30 } // end class GradeBookTest
```

Изпълнява *set* метода за courseName

Изпълнява displayMessage

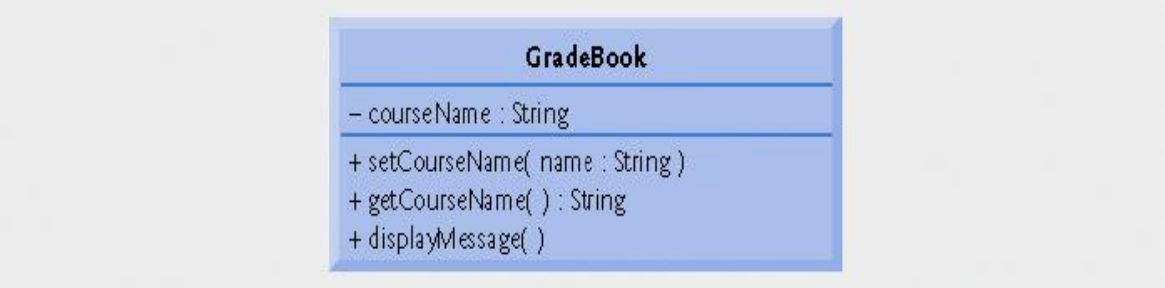
UML Class диаграма за class GradeBook

Данни на класа

- Описани в средната част
- Името на данната, следвано от две точки и типа на данната

Return тип на метод

- Указва се след името на метода с две точки, следвано от типа на данните, връщани от метода



Сравнение между примитивни и референтни типове данни

Типове данни в Java – всяка променлива има определен тип

- **Примитивни**

- boolean, byte, char, short, int, long, float, double
  -Инициализират се с конкретна стойност по подразбиране boolean се инициализира с false, всички останали взимат числено представяне на **нула**

- **Референтни**

- Референции към обекти на класове
- Получават стойност null по подразбиране(референция към „никъде”)
- Не могат да се използват, преди да се инициализират на обект от даден клас
- Позволяват извикване на методи на обект

Пример : **myGradeBook.setCourseName(theName);**

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

office@progressbg.net www.progressbg.net

Конструктори в Java

Конструктори

-Специализирани методи за създаване на обект от клас

- Дефинират се абсолютно същото име като класа, на който принадлежат.
- **Нямат** return тип и **не може да използват** командата return
- Най-често се дефинират като public
- Java извиква използването на конструктор за всеки клас
- Извикват се, когато ключовата дума new е последвана от име на клас и двойка кръгли скоби(отваряща и затваряща)

Видове конструктори

- Видове

  -Конструктор по подразбиране(default constructor) – списъкът с аргументи е празен и се създава обект с предварително зададен начален статус
  public GradeBook { …..}

  - Конструктор за общо ползване (general purpose constructor) – списъкът с аргументи съответства по брой и тип на данните на класа и служи за създаване на обект с конкретно зададен начален статус
  public GradeBook(String name) { ….}

  - Конструктор за копиране(copy constructor) – списъкът с аргументи съдържа само един елемент, референция към обект от същия клас и служи за създаване на обект със същия начален статус както реферирания обект
  public GradeBook(GradeBook someGradeBook) { …..}

GradeBook.java
(1 of 2)

```
1  // Fig. 3.10: GradeBook.java
2  // GradeBook class with a constructor to initialize the course name.
3
4  public class GradeBook
5  {
6     private String courseName; // course name for this GradeBook
7
8     // constructor initializes courseName with String supplied as argument
9     public GradeBook( String name )
10    {
11       setCourseName(name); // initializes courseName using the set method
12    } // end constructor
13
14    // method to set the course name
15    public void setCourseName( String name )
16    {  if (name != null)  // validation!!!!
17          courseName = name; // store the course name
18    } // end method setCourseName
19
20    // method to retrieve the course name
21    public String getCourseName()
22    {
23       return courseName;
24    } // end method getCourseName
```

Конструктор за общо ползване

Инициализира **ВСИЧКИ** данни на инстанцията посредством **`set`** методите им

`set` методите задължително ВАЛИДИРАТ и **присвояват само валидни стойности** на данните на инстанцията

GradeBook.java
(2 of 2)

```
25
26    // display a welcome message to the GradeBook user
27    public void displayMessage()
28    {
29       // this statement calls getCourseName to get the
30       // name of the course this GradeBook represents
31       System.out.printf( "Welcome to the grade book for\n%s!\n",
32          getCourseName() );
33    } // end method displayMessage
34
35 } // end class GradeBook
```

GradeBookTest class :

GradeBookTest.java

```
1  // Fig. 3.11: GradeBookTest.java
2  // GradeBook constructor used to specify the course name at the
3  // time each GradeBook object is created.
4
5  public class GradeBookTest
6  {
7     // main method begins program execution
8     public static void main( String args[] )
9     {
10       // create GradeBook object
11       GradeBook gradeBook1 = new GradeBook(
12          "CS101 Introduction to Java Programming" );
13       GradeBook gradeBook2 = new GradeBook(
14          "CS102 Data Structures in Java" );
15
16       // display initial value of courseName for each GradeBook
17       System.out.printf( "gradeBook1 course name is: %s\n",
18          gradeBook1.getCourseName() );
19       System.out.printf( "gradeBook2 course name is: %s\n",
20          gradeBook2.getCourseName() );
21    } // end main
22
23 } // end class GradeBookTest
```

Извиква конструктор за създаване на първия grade book обект

Създаване на втори grade book обект

```
gradeBook1 course name is: CS101 Introduction to Java Programming
gradeBook2 course name is: CS102 Data Structures in Java
```

Правила за добро програмиране

- ✓ Конструкторите се пишат веднага след декларациите на клас данните, **преди всички останали методи** на класа.
- ✓ След тях се пишат set и get методите за съответните клас данни
- ✓ После се пишат всички останали методи на класа
- ✓ Дефинирането на трите вида конструктори във всеки клас е задължително условие за правилното логическо изържано създаване на обекти в Java.
- ✓ Абсолютно задължително е всеки клас да има конструктор за общо ползване, когато класът има клас данни.

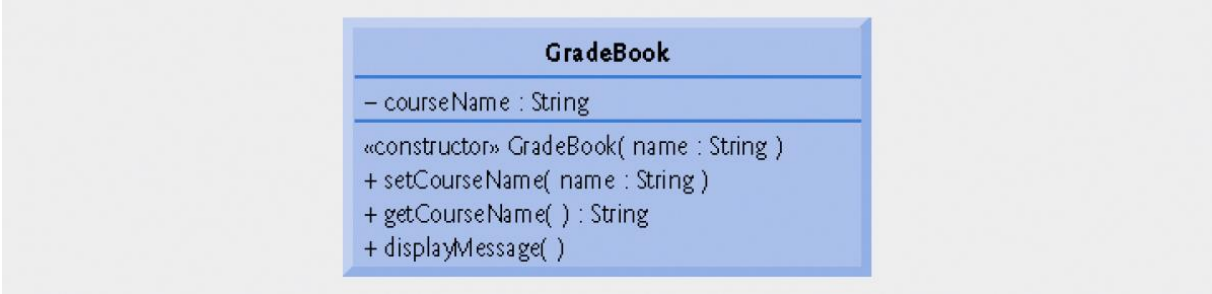
гр. София, пл. Славейков 11 ет.5 тел. : +359 897 91 93 96

office@progressbg.net www.progressbg.net

Включването на конструктор в UML Class диаграмата на класа GradeBook

-Конструкторите се описват в долната част на класа, но над всеки от методите на класа

- Вмъкнете „<<constructor>>” пред името на конструктора
- По правило, описвайте конструкторите преди всички останали методи в UML диаграмата



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!

Числа с плаваща запетая и тип данни double

Числа с плаваща запетая

- **float**

- - 32 бита,7 значещи цифри след десетичната запетая
  - константите се означават 0.5f , -2.0f

- **double**

  - 64 бита,15 значещи цифри след десетичната запетая
  - Използва двойно повече памет от float и съответно дава по-голяма точност(броя цифри след десетичната точка) при пресмятане от float
  - константите се означават **0.5d**, -**2.0d**

Обичайна грешка при програмиране

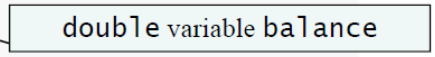
Грешка при компилация се издава винаги, когато променлива от тип с по-малка точност се присвоява от тип с по-голяма точност

Пример :

```
float f = 2.0; //грешка, 2.0 е double по подразбиране
float f = 2.0f; // правилно, 2.0 е дадена като флоат
```

Account.java

```java
// Fig. 3.13: Account.java
// Account class with a constructor to
// initialize instance variable balance.

public class Account
{
   private double balance; // instance variable that stores the balance

   // constructor
   public Account( double initialBalance )
   {
      // validate that initialBalance is greater than 0.0;
      // if it is not, balance is initialized to the default value 0.0
      if ( initialBalance > 0.0 )
         balance = initialBalance;
   } // end Account constructor

   // credit (add) an amount to the account
   public void credit( double amount )
   {
      balance = balance + amount; // add amount to balance
   } // end method credit

   // return the account balance
   public double getBalance()
   {
      return balance; // gives the value of balance to the calling method
   } // end method getBalance

} // end class Account
```



Class AccountTest

- **Форматен спесификатор %f**

  - Използва се за данни от тип плаваща запетая

  - Числото след знака за процент и f задава цифрите след десетичната точка, които ще се взимат при преобразуване на числото в текст

  Пример :
  %.5f – показа 5 цифри след десетичната запетая
  %8.2f – числото ще се представи фиксирано в 8 символни позиции с 2 цифри след десетичната запетая

```
1  // Fig. 3.14: AccountTest.java
2  // Create and manipulate an Account object.
3  import java.util.Scanner;
4
5  public class AccountTest
6  {
7     // main method begins execution of Java application
8     public static void main( String args[] )
9     {
10       Account account1 = new Account( 50.00 ); // create Account object
11       Account account2 = new Account( -7.53 ); // create Account object
12
13       // display initial balance of each object
14       System.out.printf( "account1 balance: $%.2f\n",
15          account1.getBalance() );
16       System.out.printf( "account2 balance: $%.2f\n\n",
17          account2.getBalance() );
18
```

AccountTest.java
(1 of 3)

```
19       // create Scanner to obtain input from command window
20       Scanner input = new Scanner( System.in );
21       double depositAmount; // deposit amount read from user
22
23       System.out.print( "Enter deposit amount for account1: " ); // prompt
24       depositAmount = input.nextDouble(); // obtain user input
25       System.out.printf( "\nadding %.2f to account1 balance\n\n",
26          depositAmount );
27       account1.credit( depositAmount ); // add to account1 balance
28
29       // display balances
30       System.out.printf( "account1 balance: $%.2f\n",
31          account1.getBalance() );
32       System.out.printf( "account2 balance: $%.2f\n\n",
33          account2.getBalance() );
34
35       System.out.print( "Enter deposit amount for account2: " ); // prompt
36       depositAmount = input.nextDouble(); // obtain user input
37       System.out.printf( "\nadding %.2f to account2 balance\n\n",
38          depositAmount );
39       account2.credit( depositAmount ); // add to account2 balance
40
```

Въвеждане на double стойност

Въвеждане на double стойност

AccountTest.java
(2 of 3)

```
41       // display balances
42       System.out.printf( "account1 balance: $%.2f\n",
43          account1.getBalance() );
44       System.out.printf( "account2 balance: $%.2f\n",
45          account2.getBalance() );
46    } // end main
47
48 } // end class AccountTest
```

Извеждане на double стойност

AccountTest.java
(3 of 3)

```
account1 balance: $50.00
account2 balance: $0.00

Enter deposit amount for account1: 25.53

adding 25.53 to account1 balance

account1 balance: $75.53
account2 balance: $0.00

Enter deposit amount for account2: 123.45

adding 123.45 to account2 balance

account1 balance: $75.53
account2 balance: $123.45
```

AccountTest.java

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

office@progressbg.net www.progressbg.net

UML диаграма на клас Account

Обобщение

- Всяка декларация на клас изисква ключовата дума class непосредствено преди името на класа.
- Декларация на метод, започваща с ключовата дума public указва, че е „общо достъпен” и съответно този метод може да се извиква от други класове.
- Ключовата дума void указва, че дадения метод ще изпълни действие, но няма да върне резултат от изпълнението си.
- Аргументите и променливите, декларирани в тялото на метода се наричат локални променливи
- Локалните променливи на метод са валидни само в тялото на метода, където са декларирани.
- Обектите на всеки клас се инициализират от специален метод, наречен конструктор
- Има три вида конструктори по подразбиране , за общо ползване, и конструктор за копиране
- Името на конструктора съвпада с името на класа, чиито обекти конструира
- Конструктор не може да има тип на връщане на данни, нито да използва командата return.
- Всеки клас има един или повече методи, които манипулират данните на класа.Данните описват текущото състояние на всеки обект от този клас.
- По правило данните на всеки клас са с private модификатор за достъп.

Задачи

1. Добавете в class Account метод с име debit() , който позволява да се теглят пари от обекти на class Account.Нека метода проверява дали сумата за дебит не надвишава баланса на съответния обект Account.Ако сумата за дебит надвишава този баланс, то балансът трябва да се остави без променя и метода да изведе съобщение в диалогов прозорец „Debit amount exceeded account balance.”Направете необходимите промени в class AccountTest за да тествате метода debit().

2. Променете class GradeBook така че:
- Да има втора клас данна от тип String за представяне на името на преподавателя на курса
- Напишете set метод за промяна на името на инструктора,а също и съответен get метод за намиране на текущата стойност на тази данна
- Променете конструктора за общо ползване, така че да инициализира двете клас данни – courseName и новата данна за името на преподавателя
- Променете метода displayMessage така че първо да извежда текст поздрав(welcome message) И името на курса, а след това да извежда текста „This course Is presented by :” следвано от името на преподавателя.Променете тест класа така че да демонстрира новите възможности на класа.

3.Напишете class Invoice, който може да се използва от магазин за хардуер при издаване на фактури за стока, продадена в магазина. Един обект от този клас (фактура) трябва да има 4 клас данни –

- код на компонента(part number) от тип String
- описание на компонентата(part description) от тип String
- бройки на закупената компонента(quantity) от тип int
- цена на компонентата (price) От тип double

Вашия клас трябва да има конструктор за общо ползване, който да инициализира 4-те клас данни.

Напишете set и get метод за всяка от клас данните,а за всяка от клас данните приложете концепцията за скриване на информация(data hiding)

В допълнение, напишете метод getInvoiceAmount() за пресмятане на общата сума по фактурата(бройката закупени от компонентата по нейната цена), и тогава връща тази дума като double тип данна.

Напишете отделен клас да тестване на приложението и именувайте този клас като InvoiceTest.Напишете пълен набор от команди, които да демонстрират по-горе описаниете задания като правилно изпълнени.

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

office@progressbg.net www.progressbg.net