

Лекция 2.2

Въведение в програмирането на Java.
Създаване на Java Console
приложения.
Представяне на данни в паметта.

Основни теми

- Писане на прости приложения на Java
- Използване на команди за входно изходни операции.
- Прости типове на данни в Java .
- Представяне на данни в паметта.
- Обобщение
- Задачи

- 2.1 Преговор от предишната лекция
- 2.2 Въпроси
- 2.3 Редактиране на програма на Java
- 2.4 Форматиране на изходния текст с *printf()*
- 2.5 Втора програма на Java: Събиране на цели числа
- 2.6 Структура на паметта- концепции
- 2.7 Обобщение
- 2.8 Задачи

Литература:

Java How to Program, Sixth Edition, Chapter 2

2.1 Преговор

- **Първа програма на Java**
 - коментари
 - Ключови думи
 - Дефиниране на class
 - Дефиниране на main() метод
 - Идентификатори в Java
 - Извеждане на текст на Стандартен изход
 - Правила за добър стил на програмиране

```
1 // Fig. 2.1: Wel come1.j ava
2 // Text-printing program.
3
4 public class Wel come1
5 {
6     // main method begins execution of Java application
7     public static void main( String args[] )
8     {
9         System.out.println( "Wel come to Java Programmi ng! " );
10
11     } // end method mai n
12
13 } // end class Wel come1
```

Wel come to Java Programmi ng!

2.2 Въпроси

- Напишете **името на файла**, под който ще съхраните следния клас на Java

```
public class BigClass { }
```

- **Какво** е `System.out` (клас или обект) и **какво означава** ?
- **Намерете грешките** в следната дефиниция на метода `main()`

```
public static void main(string args[])  
{  
    System.out.Println('Error!')  
}
```

2.3 Допълнения към Първата Java Program

- Променяме примерната програма Fig. 2.1 да използва различен начин за извеждане на текст
- Команди

`print()`

`println()`

Outline

Wel come2. j ava

1. Comments
2. Blank line
3. Begin class
Wel come2

System.out.print keeps the cursor on the same line, so System.out.println continues on the same line.

System.out.print

4.1 Method
System.out.print
ln

5. end main,
Wel come2

Program Output



Е. Кръстев, Увод в
Програмирането
2012

```
1 // Fig. 2.3: Wel come2. j ava
2 // Printing a line of text with multiple statements.
3
4 public class Wel come2
5 {
6     // main method begins execution of Java application
7     public static void main( String args[] )
8     {
9         System.out.print( "Wel come to " );
10        System.out.println( "Java Programmi ng!" );
11
12    } // end method main
13
14 } // end class Wel come2
```

Wel come to Java Programmi ng!

2.3 Допълнения към ...

Промени

- **Welcome2.java** (Fig. 2.3) извежда същия текст като **Welcome1.java** (Fig. 2.1)
- Използва следните команди

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

- Line 9 извежда “Welcome to ” като **курсора остава на същия ред**
- Line 10 дописва “Java Programming! ” на същия ред като **курсора се премества на следващия ред**

2.3 Допълнения към ...

- **Escape characters**
 - Backslash (\)
 - Интерпретира извеждане на специални символи
- **Символ за нов ред (\n)**
 - Interpreted as “special characters” by methods `System.out.print` and `System.out.println`
 - Премества курсора на следващия ред
 - `Welcome3.java` (Fig. 2.4)

```
9      System.out.println( "Welcome\nto\nJava\nProgramming! " );
```

- Ред от текст се пренася на следващия ред при извеждане на \n

Outline

Wel come3. j ava

1. mai n

2.
System.out.println
(uses \n for new
line)

Program Output

```
1 // Fig. 2. 4: Wel come3. java
2 // Printing multiple lines of text with a single statement.
3
4 public class Wel come3
5 {
6     // main method begins execution of Java application
7     public static void main( String args[] )
8     {
9         System.out.println( "Wel come\nJava\nProgrammi ng!" );
10
11     } // end method mai n
12
13 } // end class Wel come3
```

Wel come
to
Java
Programmi ng!

Notice how a new line is output for each \n escape sequence.



2.3 Описания на действието на специалните символи ...

Escape sequence	Description
<code>\n</code>	Newline. Курсорът се премества на следващия ред.
<code>\t</code>	Horizontal tab. Премества курсора на следващият TAB.
<code>\r</code>	Carriage return. Връща курсора в началото на текущия ред , не се премества на следващия ред.
<code>\\</code>	Backslash. Извежда символа <code>\</code> без да използва специалното му значение .
<code>\"</code>	Double quote. Извежда <code>"</code> . For example, <pre>System.out.println("\"i n quotes\"");</pre> отпечатва <pre>"i n quotes"</pre>

Fig. 2.5 | Описания на действието на специалните символи .

2.4 Форматиране на текст с `printf`

- `System.out.printf`
 - Нововъведение от J2SE 5.0
 - Форматира изходния текст- пример

```
9      System.out.printf( "%s\n%s\n",  
10         "Wel come to", "Java Programmi ng!" );
```

Синтаксис `printf(formatString, argList);`

- `formatString` включва
 - Фиксиран текст
 - Форматни спесификатори – **placeholder** (място за вмъкване на стойност)
 - Пример `%s` – placeholder за низ от символи `String`
- `argList` включва списък от идентификатори, които се съпоставят на форматните спесификатори

Outline

Wel come4. j ava

mai n

printf

System.out.printf
displays formatted data.

Program output



```
1 // Fig. 2.6: Wel come4. j ava
2 // Printing multiple lines in a dialog box.
3
4 public class Wel come4
5 {
6     // main method begins execution of Java application
7     public static void main( String args[] )
8     {
9         System.out.printf( "%s\n%s\n",
10             "Wel come to", "Java Programmi ng!" );
11
12     } // end method main
13
14 } // end class Wel come4
```

Wel come to
Java Programmi ng!

Правила за добро програмиране 2.9

Оставяйте по един празен символ след запитая (,) в списъка от аргументи за по-добра читаемост на програмата.

Обичайна грешка при програмиране 2.7

Не се разрешава пренасяне на команда или текстова константа между два или повече реда.

```
// Това е ГРЕШНО!
```

```
9      System.out.printf( "%s\n%s\n  
10      Wel come to", "Java Programmi ng! " );
```

```
// .... и това е ГРЕШНО!
```

```
9      System.out.  
10      printf( "%s\n%s\n",  
11      "Wel come to", "Java Programmi ng! " );
```


2.5 Събиране на цели числа

- Следваща програма

- Използва клас **Scanner** за въвеждане на две цели числа от Стандартен вход
- Използва **printf()** да изведе сумата на тези числа на Стандартен изход
- Демонстрира използване на библиотека на Java-**package**

Outline

Addi ti on. j ava

(1 of 2)

```

1 // Fig. 2.7: Addi ti on. java
2 // Addition program that displays the sum of two numbers.
3 import java.util.Scanner; // program uses class Scanner
4
5 public class Addi ti on
6 {
7     // main method begins execution of Java application
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        int number1; // first number to add
14        int number2; // second number to add
15        int sum; // sum of number1 and number2
16
17        System.out.print( "Enter first integer: " ); // prompt
18        number1 = input.nextInt(); // read first number from user
19

```

import декларация за импорт на class **Scanner** от библиотека **java.util**.

Декларира и инициализира променлива **input**, която е Scanner **обект**.

Декларира променливи **number1**, **number2** и **sum**.

Прочита цяло число от Стандартен вход и го присвоява на **number1**.



Outline

```

20 System.out.print( "Enter second integer: " ); // prompt
21 number2 = input.nextInt(); // read second number from user
22
23 sum = number1 + number2; // add numbers
24
25 System.out.printf( "Sum is %d\n", sum ); //
26
27 } // end method main
28
29 } // end class Addition

```

Прочита цяло число от Стандартен вход и го присвоява на **number2**.

Пресмята сумата на променливите **number1** и **number2**, присвоява резултата на **sum**.

Извежда **sum** с форматиране.

```

Enter first integer: 45
Enter second integer: 72
Sum is 117

```

Двете цели числа въведени от потребителя.

Addition.java

(2 of 2)

4. Addition

5. printf



2.5 Събиране на цели числа

```
3  import java.util.Scanner;    // program uses class Scanner
```

– **import** декларации

- Използват се от компилатора да разпознае и намери дефинициите за класове в Java programs
- Указва на компилатора да прочете дефиницията на class **Scanner** от **java.util** package

```
5  public class Addition  
6  {
```

– Началото на **public class Addition**

- Името на файла с този клас задължително е **Addition.java**

– Редове 8-9: начало на **main()**

Обичайна грешка при програмиране 2.8

Всички `import` декларации трябва да се пишат в група **ПРЕДИ** декларацията на първия клас във файла.

Поставяне на `import` декларация във вътрешността на клас или след тялото на класа е **`syntax error`**.

Обичайна грешка при програмиране 2.7

Пропускането на **import** декларация за клас от Java библиотека води до грешка при компилация със съобщение указващо “*cannot resolve symbol*.” При такава грешка проверете, че подходящите **import** декларации са указани, а също и имената на **import** декларациите са правилно написани, особено големите и малки букви.

2.5 Събиране на цели числа

```
10 // create Scanner to obtain input from command window
11 Scanner input = new Scanner( System.in );
```

– Променливи

- Място в паметта съхраняващо стойност от даден тип
 - Името и типа на променливата се **декларират преди** да се използват
- **input** е от тип **Scanner**
 - Позволява да се четат данни от Стандартен вход
- Име на променлива: може да е всеки идентификатор

– Всяка декларация завършва с **;**

– Променлива може да се инициализира при декларирането ѝ

- Присвояване- използва знак за равенство
- Обект за достъп до Стандартен вход (**in** е клас обект на **class System**)
 - **System.in**
 - Поддържа методи за четене от Стандартен вход

2.5 Събиране на цели числа

```
13      int number1; // first number to add
14      int number2; // second number to add
15      int sum;      // second number to add
```

- Декларация на `number1`, `number2` и `sum` от тип `int`
 - `int` означава **integer стойности** (whole numbers): i.e., 0, -4, 97
 - типове `float` и `double` означават числа с **плаваща запетая**
 - тип `char` съхранява символ : i.e., `'x'`, `'$'`, `'\n'`, `'7'`
 - `int`, `float`, `double` и `char` са прости данни
- Хубаво е да се използват коментари за предназначението на променливите

```
int number1, // first number to add
    number2, // second number to add
    sum      ; // second number to add
```

- Една декларация позволява да се декларира повече от една променлива
- Използва се списък от променливи разделени със запетая
- **ВСЯКА ДЕКЛАРАЦИЯ ЗАВЪРШВА С ;**

Правила за добро програмиране 2.10

Декларирайте всяка променлива на отделен ред.

Това придава по-добро описание и читаемост на програмата и лесна промяна в описанието на променливите.

Правила за добро програмиране 2.11

Изборът на смислени имена придава на програмата само- описателен характер *self-documenting* (т.е програмата може да се разбере като се чете кода ѝ, вместо да се използва специално ръководство - manual).

Правила за добро програмиране 2.12

По подразбиране (и задължително по време на този курс!), имената на променливите започват с малка буква, и всяка дума след първата започва с главна буква. Например, променливата `firstNumber`.

2.5 Събиране на цели числа

```
17      System.out.print( "Enter first integer: " ); // prompt
```

- *Message called a prompt* – Съобщение- текст подканящ потребителя да въведе данни
- Библиотека- **package java.lang**

```
18      number1 = input.nextInt(); // read first number from user
```

- Резултатът от `nextInt()` се присвоява на `number1` с помощта на оператора **=**
 - Инstrukция за присвояване
 - **= бинарен оператор** – използва два операнда
 - Изразът от дясно се пресмята и присвоява на променливата отляво
 - Чети като:

`number1` приема стойността на `input.nextInt()`

Забележка 2.1

Библиотеката, package `java.lang` се импортира във всяко приложение по подразбиране- не трябва явно да се прави импорт на тази библиотеката.

`java.lang` е единствената библиотека на Java API за която не се изисква `import` декларация.

Правила за добро програмиране 2.13

Оставяйте празен символ от двете страни на бинарния оператор, за да придадете читаем вид на програмата си.

2.5 Събиране на цели числа

```
20      System.out.print( "Enter second integer: " ); // prompt
```

- Аналогично на предната инструкция
 - Подканва потребителя да въведе следващото цяло число

```
21      number2 = input.nextInt(); // read second number from user
```

- Аналогично на предната инструкция
 - Присвоява второто въведено цяло число на променливата `number2`

```
23      sum = number1 + number2; // add numbers
```

- Инструкция за присвояване
 - Пресмята сумата на `number1` и `number2` (от дясно на ляво)
 - Използва оператор за присвояване `=` да присвои резултата от събирането на променливата `sum`
 - Чети като: `sum` получава стойността на `number1 + number2`
 - `number1` и `number2` са операнди

2.5 Събиране на цели числа

```
25      System.out.printf( "Sum is %d \n: " , sum ); // display sum
```

- Използва **System.out.printf** да изведе резултата
- Използва форматиращ спецификатор **%d**
 - Указва място за вмъкване на **int** стойност от списъка с аргументи на **printf**

```
System.out.printf( "Sum is %d\n: " , ( number1 + number2 ) );
```

- Пресмятанията могат да се извършат в списъка с аргументи на **printf**
- Скобите около **number1 + number2** не са задължителни- използват се за по- голяма яснота на кода

2.6 Структура на паметта- концепции

- **Променливи**

- Всяка променлива има
 - **ИМЕ,**
 - **ТИП,**
 - **РАЗМЕРНОСТ и**
 - **СТОЙНОСТ**
- Името символизира адреса в паметта заеман от променливата
- Когато **присвояваме/ записваме** нова стойност на променлива, тя **замества/ изтрива** предишната стойност на променливата
- Четене на променливи от паметта **не променя адреса или стойността им**

2.6 Структура на паметта- концепции

- Деклариране на променлива- задава:
 - Всяка променлива има
 - ИМЕ,
 - ТИП,
 - РАЗМЕРНОСТ и **евентуално**
 - СТОЙНОСТ
 - При изпълнение на програмата, **всяка декларация на променлива от прост тип** води до заделяне на област в паметта в съответствие с **ТИПА и РАЗМЕРНОСТТА** на променливата
- Прости типове променлива

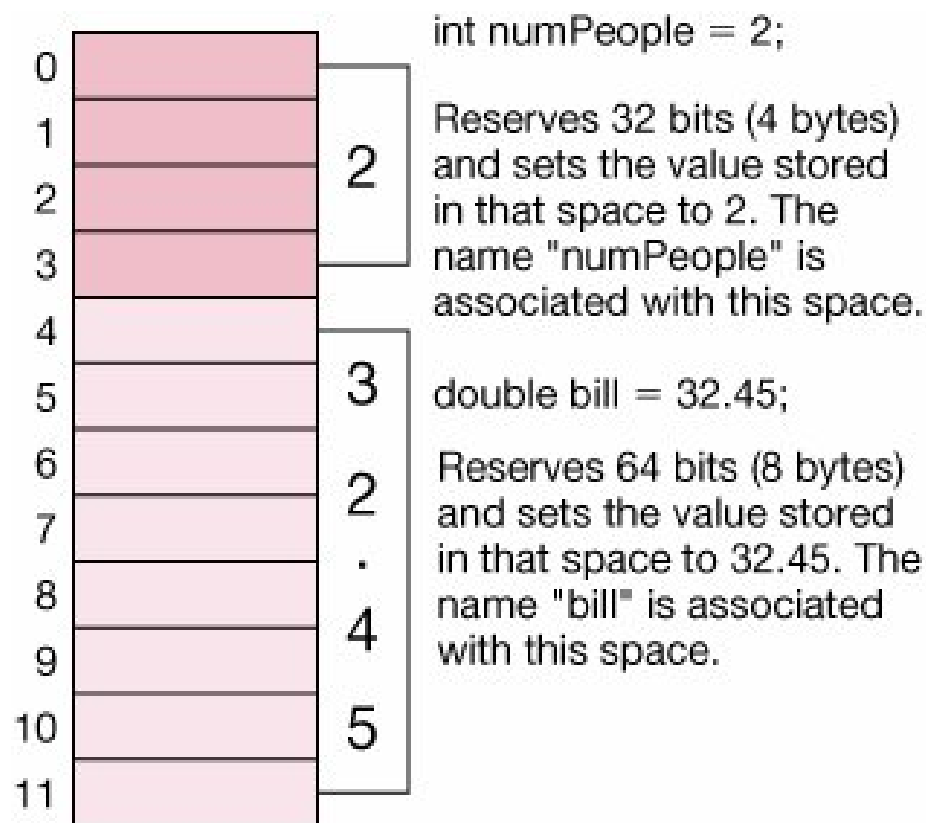
`byte, short, int, long, float,
double, boolean` **или** `char`

2.6 Структура на паметта- концепции

Примери за декларации на променливи от прост тип:

```
{  
    int numPeople = 2;          //заделя 2 байта  
    double bill    = 32.45;    //заделя 4 байта  
    double tip     = bill * 0.2;  
    double total   = bill + tip;  
    double totalPerPerson = total / numPeople ;  
  
    System.out.println(numPeople) ;  
    System.out.println(bill) ;  
    System.out.println(tip) ;  
    System.out.println(total) ;  
    System.out.println(totalPerPerson) ;  
}
```

2.6 Структура на паметта- концепции



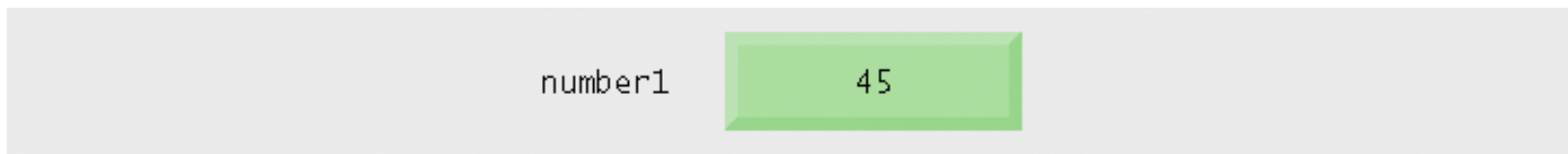


Fig. 2.8 | Адрес в паметта, представляван от променливата `number1` и стойността на тази променлива съхранявана на този адрес.

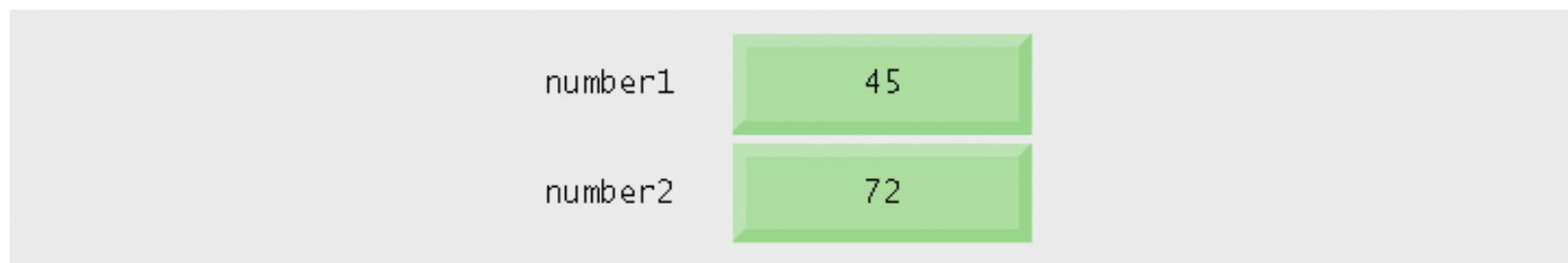


Fig. 2.9 | Адресите в паметта и стойностите на променливите `number1` и `number2` след присвояването им на стойности 45 и 72.

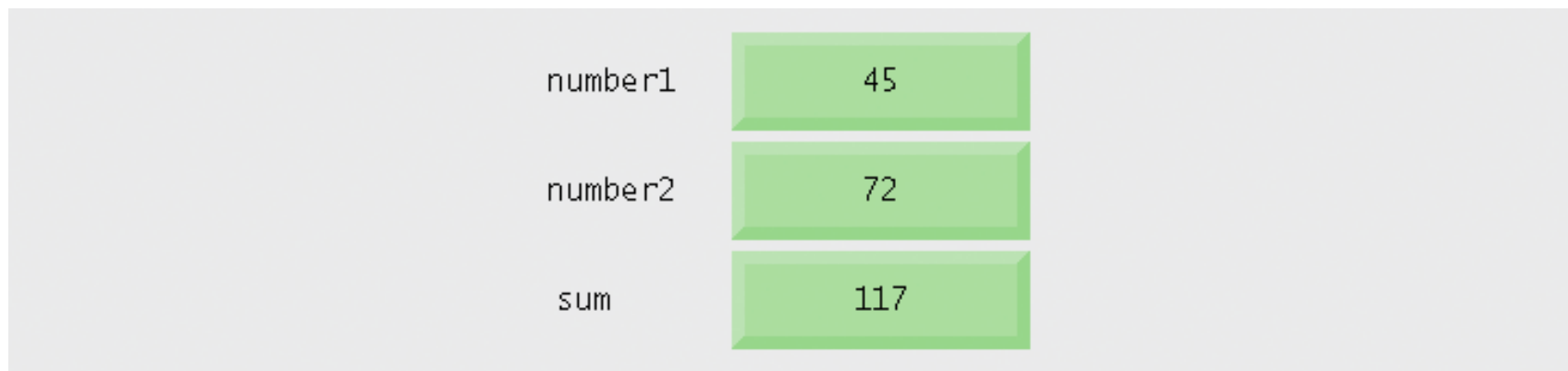


Fig. 2.10 | Адресите в паметта и стойностите след пресмятане и присвояване на сумата `number1` и `number2` в променливата `sum`.

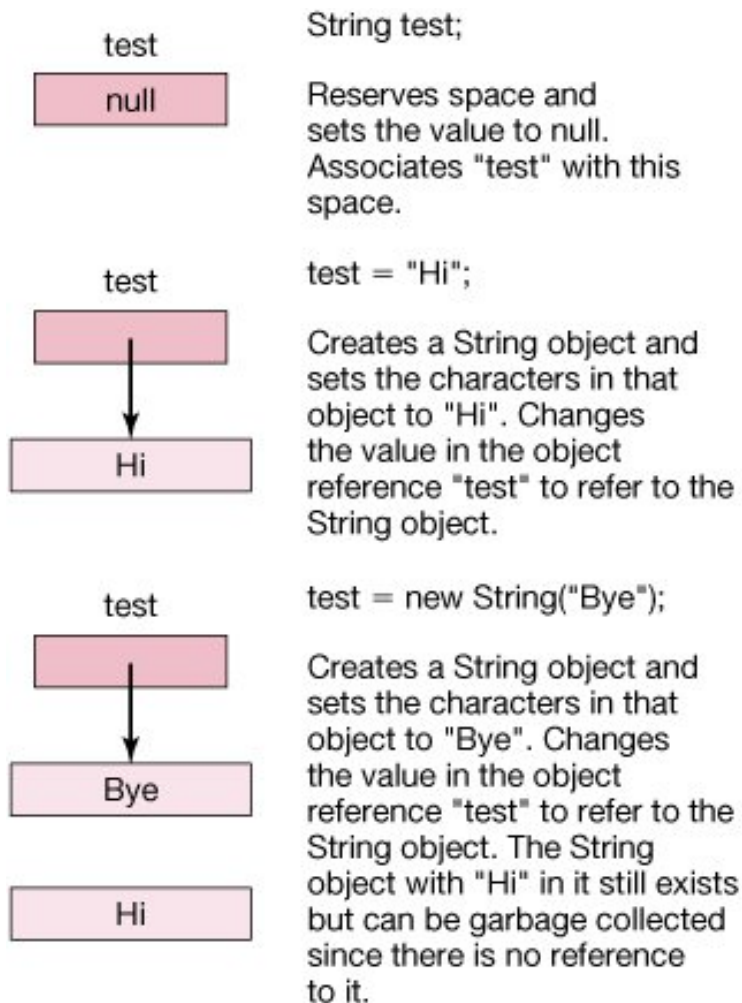
2.6 Структура на паметта- концепции

- **Типове данни**
 - **Примитивни** типове данни
`byte, short, int, long, float, double, boolean` или `char`
запазват памет (*стека* на процеса) и асоциират името на променливата с адреса на началото на така отделената област от паметта.
 - Променливи **от всеки друг тип** се наричат **референтен тип данни**. Това са променливи, които взимат за стойност обект от клас
- Референтни типове данни се **характеризират** с
 - ИМЕ – **идентификатор** на Java
 - ТИП - **име на клас**
 - РАЗМЕРНОСТ - размерността на обекта от съответния клас
 - СТОЙНОСТ - **обект от клас**
- Стойността по подразбиране на променлива от референтен тип данни е `null`

ПРИМЕР:

```
Scanner input;
// декларация на променлива от референтен тип Scanner
input = new Scanner(System.in); // задаване на стойност
String test;
test = "Hi";
test = new String("Bye");
```


2.6 Структура на паметта- концепции



Обичайна грешка при програмиране 2.7

Всяка променлива трябва да се инициализира, **преди** да се използва в израз за присвояване.

Това е задължително за референтните типове данни.

Простите типове данни се инициализират по подразбиране, но **въпреки това е препоръчително това правило да се спазва и за тях с оглед избягване на логически грешки**

2.8 Обобщение

- Името на всеки Java class е идентификатор, който е поредица от букви, цифри, подчертаване (`_`) и знак (`$`) и не започва с цифра и не съдържа празни символи
- Името на идентификатор не може да съвпада с ключова дума на Java.
- Java е зависима от малки и големи букви в идентификаторите.
- Тялото на всеки клас декларация се ограничава със
Методът `main()` е начална точка за изпълнение на всяко Java приложение и започва с

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

В противен случай JVM няма да изпълни приложението.

2.9 Задачи

1. Напишете приложение, което извежда числата 1 до 4 на един ред, разделени със запетая и шпация след нея. Използвайте следните техники:
 - a) една *`System.out.println`* команда.
 - b) четири *`System.out.print`* команди.
 - c) една *`System.out.printf`* команда.
2. Напишете Java приложение, което чете *цяло число*, определя дали е *четно* или *нечетно* и *извежда на стандартния изход* съобщение на получения резултат

2.9 Задачи

3. Напишете приложение, имплементира следния алгоритъм:

Стъпка 1. Прочетете цялото число N от клавиатурата.

Стъпка 2. Отпечатайте N .

Стъпка 3. Ако N е четно разделете го на 2.

Стъпка 4. Ако N е нечетно умножете N по 3 и прибавете 1

Стъпка 5. Отпечатайте N .