Технически университет – София
Факултет по Компютърни системи и управление

ПИК 3 - Java

Семинарни упражнения

гл. ас. д-р Антония Ташева

Теми

1. Въведение в Java, типове данни, масиви и оператори в Java, оператори if-else, switch, цикли, стандартен вход и изход, вход и изход - работа с файлове.
2. Изключения, методи, хвърляне на изключения, обект String, динамични низове (StringBuffer), регулярни изрази.
3. Защита на домашни задания от седмици 1 и 2.
4. Класове и пакети, наследяване, абстрактни класове и интерфейси, полиморфизъм, капсулация на данни.
5. Защита на домашни задания от седмица 4.
6. Константи, под-пакети и не-статични вложени член класове, локални класове в Java, анонимни класове в Java, статични вложени член класове, Garbage Collection и метод finalize()
7. Контролна работа №1.

Теми

8. Списъци, динамични масиви, стекове и опашки, изключения дефинирани от потребителя.
9. Нишки, синхронизация на нишки, решаване на задача по предварително зададено приложение.
10. Защита на домашни задания за седмици 8 и 9.
11. Сокети, работа по примерно приложение работещо със сокети.
12. Графична среда, създаване на елементарни приложения със Swing.
13. Защита на домашни задания от седмици 11 и 12. Заверяване на семестъра.
14. Контролна работа №2.

Оценяване и заверки

Компонент	Периодичност	Брой	Точки
1. Участие в лабораторно упражнение	Всяка седмица без защити и контролни	8 упр.	2 точки за присъствие на упражнение
2. Успешна защита на домашно задание	Четири пъти, разпределено равномерно през семестъра	4 домашни задания	6 точки на задание
3. Контролна работа 1	Еднократно, по време на лабораторно упражнение	1	20 точки
4. Контролна работа 2	Еднократно, по време на лекция	1	40 точки

- Оценка среден (3) за 61 точки и нагоре
- Заверка за поне 24 точки от 1, 2 и 3

Допълнителна информация за курса

- Сайт на проф. Даниела Гоцева
  dgotseva.com
- Блог на ас. Филип Петров
  www.cphpvb.net
- Сайт на катедра „Компютърни системи“
  cs.tu-sofia.bg
- Сайт на гл. ас. Антония Ташева
  tasheva.info

типове данни,
масиви и оператори в Java,
оператори if-else, switch, цикли,
стандартен вход и изход,
работа с файлове

ТЕМА 1. ВЪВЕДЕНИЕ В JAVA

Какво е  ?

- Java е език за програмиране създаден през 1991г. (първоначално под името Oak, а след 1995г. Java).
- Основите на езика идват от езиците C и C++.
- Синтаксисът на езика е много близък до C,
- Обектния модел е изключително близък до C++.

Какво е Java?

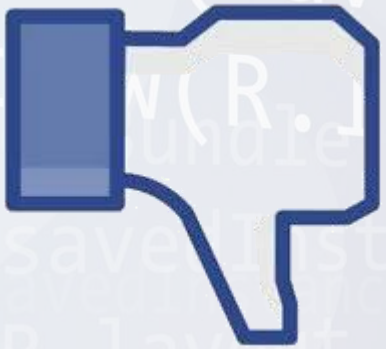
- Един от най-широко разпространените езици за програмиране.
- Проектиран е да бъде малък, прост и преносим между различни платформи и операционни системи на ниво код и на бинарно ниво.
- Java програми, които включват аплети или приложения могат да работят само на машини с инсталиран Java Virtual Machine (JVM).

Предимства

- *Java е лесен за учене*
- *Java е обектно-ориентиран език*
  Това позволява създаването на модулни приложения и преизползваем код.
- *Java е лек*
  Позволява писането на програми за устройства с ограничени ресурси
- *Има възможност за многонишково изпълнение*

Недостатъци

- Java се счита за по-бавен и често може да използва повече памет от другите езици като C и C++.
- GUI с Java...



Java е обектно-ориентиран език за програмиране ?!

- ***Абстракция*** – обозначаване на характеристики, които отличават обектите един от друг.
- ***Капсулиране*** – скриване на информацията за конкретната програмна реализация на даден обект и показване само на тези негови части, които влизат в практическа употреба за останалите обекти;
- ***Полиморфизъм*** – възможност за извършване на функционално различни действия с едни и същи инструменти;
- ***Наследяване*** – процес, при който един обект приема и разширява свойствата на друг.

Какво ни трябва за да започнем да пишем на Java?

- „компилаторът“, който превръща код на Java до байткод - Java Development Kit (JDK).
- „докомпилиране“ на байт код до изпълним - Java Runtime Environment (JRE), която е част от JDK
- среда за разработка на програмите – налични са множество безплатни продукти, например Eclipse, Netbeans и DrJava.

Файлове с код

- „Сорс-кодът“ на програмите на Java се записва във файлове с разширение .java.
- Когато бъдат компилирани до байткод те получават разширение .class. Именно този .class файл е преносимият код, който може да бъде компилиран на различни машини.

myfirstprogram

- Във файла .java дефинираме клас със същото име като файла и той да бъде публичен, т.е. „достъпен за външния свят“.
- В този клас се дефинира и основен метод (в Java е прието функциите и процедурите да се наричат методи) с име „main“.
- Създайте текстов файл с име myfirstprogram.java и запишете в него следната информация:

```
public class myfirstprogram{
  public static void main(String[] args) {
    System.out.format("%s", "Hello World\n");
  }
}
```

Типове данни

- boolean – булев тип приемащ за стойност литерали true или false;
- char – символ в кодировка unicode;
- byte – 8 битово цяло число;
- short – 16 битово цяло число;
- int – 32 битово цяло число;
- long – 64 битово цяло число;
- float – 32 битово число с плаваща запетая;
- double – 64 битово число с плаваща запетая.

- String - Той се състои от поредица от 16 битови unicode символи (т.е. естествена алтернатива на масив от тип char). За разлика от основните типове данни обаче, той е обект.

Специални char символи:

- \n – нов ред;
- \r – връщане в началото на ред;
- \t – табулация;
- \b – връщане на символ назад (backspace);
- \’ – апостроф;
- \“ – кавички;
- \\ – знак наклонена черта.

Масиви

- Масиви в Java могат да бъдат задавани („дефиниция на масив“) по два еквивалентни начина:

```
int[] array1;  - препоръчителен
int array2[];
```

- Масивите в Java са обекти -те се съхраняват в динамичната памет (heap) за разлика от примитивните типове, които се записват в стек (stack).
- „създаване на масив“ - Тъй като масивите са обекти, то те се създават чрез оператор new:

```
array1 = new int[10];
array2 = new int[5];
```

- „инициализация“ на масив

```
array1[0] = 4; array1[1] = 5;
```

Двумерните и многомерни масиви

- Двумерните и „n“-мерните масиви се дефинират аналогично:

```
int[][] array3 = new int[2][3];
array3[0][2] = 5;
```

Оператори в Java - Унарни

a) ++ – инкрементиране с 1;
b) – – декрементиране с 1;
c) + – положително число, например +3;
d) – – отрицателно число, например -3;
e) ~ – побитово инвертиране, например 00001111 става 11110000;
f) ! – логическо отрицание;
g) (<тип>) – преобразуване на тип данни, например ако x е от тип int, то можем да направим double d = (double)x.

Оператори в Java - Аритметични

a) + – събиране (използва се и за конкатенация на обекти от тип String);
b) – – изваждане;
c) * – умножение;
d) / – деление;
e) % – остатък от деление при целочислените типове.

Оператори в Java - Сравнение

a) < – по-малко;

b) <= – по-малко или равно;

c) > – по-голямо;

d) >= – по-голямо или равно;

e) == – равно;

f) != – не равно.

Оператори в Java - Логически

a) && – логическо „и“;
b) || – логическо „или“.

Оператори в Java - Побитови

a) & – побитово „и“;
b) | – побитово „или“;
c) ^ – побитово „изключващо или“;
d) << – изместване вляво;
e) >> – изместване вдясно;
f) >>> – изместване вдясно без знак (нов оператор, който не присъства в C/C++, но на който няма да се спираме)

Конструкция if-else

```
int a=2, b=3;
if (a!=0) System.out.format("x = %d\n",(-b/a));
else{
  if (b == 0) System.out.format("%s", "Any x is a solution");
  else System.out.format("%s", "There is no solution");
}
```

Оператор switch

```
int month = 3;
switch (month) {
  case 1:  System.out.format("%s", "January\n");
           break;
  case 2:  System.out.format("%s", "February\n");
           break;
  case 3:  System.out.format("%s", "March\n");      break;
  case 4:  System.out.format("%s", "April\n");      break;
...
  case 10: System.out.format("%s", "October\n");    break;
  case 11: System.out.format("%s", "November\n");   break;
  case 12: System.out.format("%s", "December\n");   break;
  default: System.out.format("%s", "Invalid month.\n");
}
```

Цикъл while

```
  int A = 28;
  int B = 48;
  System.out.format("NOD(%d,%d) = ", A,B);
  while (A != B){
    if (A > B)  A = (A-B);
    else  B = (B-A);
  }
  System.out.format("%d",A);
}
```

Цикъл do-while

```
int n = 6;
System.out.format("n = %d =>",n);
long factorial = 1L;
do {
  factorial *= (long)n;
  n--;
}while (n > 0);
System.out.format("n! = %d\n",factorial);
```

Цикъл for

```
int[] array = {2,5,12,4,3,13,18,-5,-4,8};
for (int i=0; i<array.length; i++){
  if (array[i]%2 == 0)
            System.out.format("%d ",array[i]);
  }
System.out.format("\n");
```

Оператор break

```
int[] array = {2,5,12,4,3,13,18,-5,-4,8};
boolean found = false;    int search = 13;
System.out.format("Searching for %d",search);
for (int i=0; i<array.length; i++){
  if (array[i] == search){
    found = true;
    break;
  }
  System.out.format("%d... ", array[i]);
}
if(found) System.out.format("FOUND>>>%d<<<FOUND!",search);
   else System.out.format("no %d in the array",search);
```

Оператор continue

```
int[] array = {2,5,12,4,3,13,18,-5,-4,8};
long sum = 0L;
for (int i=0; i<array.length; i++){
    if (array[i]%2 == 0) continue;
    sum += (long)array[i];
}
System.out.format("Sum of odd numbers = %d\n", sum);
```

Цикъл for-each

```
<тип>[] array = {стойности};
for (<тип> var : array) {
    оператори;
}
```

Аналогично на:

```
<тип>[] array = {стойности};
for (int i=0; i<array.length; i++) {
  <тип> var = array[i]
  оператори;
}
```

Цикъл for-each (Пример)

```
int[] array = {2,5,12,4,3,13,18,-5,-4,8};
for (int i : array){
  System.out.format("%d ",i);
}
System.out.format("\n");
```

Стандартен изход

- System.out.format()
- System.out.print() и System.out.println()

```
int i=2;
double d=3;
System.out.print("i = "+i+", and d = "+d+"\n");

System.out.println("Result: "+2+3);
System.out.println("Result: "+(2+3));
```

Стандартен изход за грешки

- System.err
- Повече при „обработка на изключения“

Стандартен вход

- System.in.? работи се с байтове
- System.in.read() – четене на информация байт по байт (но се записва в int). Стойност -1 показва грешка!

```
int input;   System.out.print("Enter text: ");
do{
  try{
    input = System.in.read();
    System.out.print((char)input);
  }
  catch(java.io.IOException e){
    System.err.println("ERROR READING");     break;
  }
}while(input!=(int)'\n');
```

Входен буфер

- System.in.available() връща int с броя байтове, които са налични в буфера
- System.in.skip() пък приема число от тип long, с което се пропускат определен брой байтове (т.нар. flush).

Форматиран вход

- стандартен обект – Scanner

```
java.util.Scanner s = null;
String input;
s = new java.util.Scanner(System.in);
System.out.print("Enter text: ");
input = s.next();
System.out.print(input);
```

- Пример 2

```
String input = "Hello 123 World";
java.util.Scanner s = new java.util.Scanner(input);
String s1 = s.next();
int i = s.nextInt();
String s2 = s.next();
System.out.println(s1+s2+i);
s.close();
```

Форматиран вход

- nextBoolean(), nextByte(), nextShort(), nextLong(), nextFloat() и nextDouble()
- skip().
- nextLine()
- hasNextInt(), hasNextDouble()

```
String input = "9,-12,4,a,32";
java.util.Scanner s = new java.util.Scanner(input);
s.useDelimiter(",");
while (s.hasNextInt()) System.out.print(s.nextInt()+" ");
s.close();
```

Работа с файлове

```
java.io.FileInputStream in = null;
java.io.FileOutputStream out = null;
try {
  in = new java.io.FileInputStream("input.txt");
  out = new java.io.FileOutputStream("output.txt");
  int readbyte;
  while ((readbyte = in.read()) != -1) {
    out.write(readbyte);
  }
}
```

```
catch(java.io.IOException e){
  System.err.println("Error reading file");
}
finally{
  try{
    if (in != null) in.close();
    if (out != null) out.close();
  }
  catch(java.io.IOException e){}
}
```

Класове FileInputStream и FileOutputStream - главно за бинарни файлове (8 бита)

Работа с файлове

```
java.io.FileReader in = null;
java.io.FileWriter out = null;
try {
  in = new java.io.FileReader("input.txt");
  out = new java.io.FileWriter("output.txt");
  int readbyte;

  while ((readbyte = in.read()) != -1) {
    out.write(readbyte);
  }
}
```

```
catch(java.io.IOException e){
  System.err.println("Error reading file");
}
finally{
  try{
    if (in != null) in.close();
    if (out != null) out.close();
  }
  catch(java.io.IOException e){}
}
```

Класове FileReader и FileWriter – за четене на символни файлове

Чете 16 бита

Работа с файлове

- смесени типове потоци - InputStreamReader и OutputStreamWriter („небуферирани“ потоци)
- Буферирани: **BufferedReader** и **BufferedWriter,** BufferedInputStream и BufferedOutputStream

„потоци от данни“(„Data Streams“)

- DataInputStream / DataOutputStream

```
java.io.DataInputStream in = new java.io.DataInputStream(
                    new java.io.BufferedInputStream(
                      new java.io.FileInputStream("output.txt")
```

- in.readInt()
- in.readDouble()
- in.readUTF()

PrintStream

- Има функциите print() и println()

```
java.io.PrintStream out =
        new java.io.PrintStream(
new java.io.FileOutputStream("output.txt"));

int i = 123;
out.print("Hello");
out.println(" world "+i);
```

Пренасочване на вход/изход

- Вместо в конзолата печатаме във файл:

```
try {
   System.setOut(new java.io.PrintStream(
                 new java.io.FileOutputStream("output.txt")
                  ));
    }
    catch (java.io.FileNotFoundException e) {
            System.err.println("FileNotFound");
            return;
    }
    System.out.println("Tozi text shte se zapishe vav
file");
    System.out.close();
```

изключения,
методи,
хвърляне на изключения,
обект String,
динамични низове (StringBuffer),
регулярни изрази

ТЕМА 2. ИЗКЛЮЧЕНИЯ И ДР.

Методи

- В ООП езиците функциите и процедурите се наричат методи.
- Главния метод за всяка програма е методът main.
- За сега ще разглеждаме само публични (public) и статични (static) методи
- Декларацията на метод следва следната схема:

```
public static <тип> <име>(<входни параметри>){...}
```

Пример за метод

```
public static int getLargerNum(int a, int b){
  if (a > b) return a;
  else return b;
}
public static void main(String[] args) {
  int x = 2;
  int y = 3;
  System.out.println(getLargerNum(x,y));
}
```

В методите параметрите представляват локални копия и ако бъдат променени, това няма да се отрази в извикващия метод.

Предаване на параметър по адрес

- Предава се „адреса“ на обект...

```
public static void incrementArr(int[] a){
  for (int i=0; i<a.length; i++){
    a[i]++;
  }
}
public static void main(String[] args) {
  int[] arr = {1, 3, 4, 8};
  incrementArr(arr);
  for (int i: arr){
    System.out.print(i+" ");
  }
}
```

Предаване на променлив брой параметри

```
public static void showDoubledNums(int ... nums){
   for (int i: nums){
    System.out.print((2*i)+" ");
  }
  System.out.println();
}
public static void main(String[] args) {
  showDoubledNums(1,2,5);
  showDoubledNums(12,1,14,2,4);
}
```

Единственото важно правило при подаването на променлив брой параметри е, че те трябва да са последните в списъка на входните параметри.

Връщане на резултат - масив

```
public static int[] getDoubledNums(int[] nums){
  int[] doubledNums = new int[nums.length];
  for(int i=0; i<nums.length; i++){
    doubledNums[i] = 2*nums[i];
  }
  return doubledNums;
}
public static void main(String[] args) {
  int[] x = {12,1,14,2,4};
  int[] doubledX = getDoubledNums(x);
  for(int i=0; i<x.length; i++){
    System.out.println("x = "+x[i]+", and 2*x = "+doubledX[i]);
  }
}
```

Полиморфизъм

- Полиморфизъм е свойството на един метод да прави различни операции в зависимост от обекта, който го е извикал.
- Това се осъществява като дефинираме методи с едни и същи имена, но различни входни параметри.

Полиморфизъм - пример

```
public static boolean isLarger(int a, int b){
  if (a > b) return true;
  else return false;
}
public static boolean isLarger(double a, double b){
  if (a > b) return true;
  else return false;
}
public static void main(String[] args) {
  int x1 = 2;          int x2 = 3;
  System.out.println(isLarger(x1,x2));
  double d1 = 5;       double d2 = 1;
  System.out.println(isLarger(d1,d2));
}
```

Какво е изключение?

- Изключението (exception) е събитие, което не е предвидено за нормалното изпълнение на програмата.
- Изключенията дават унифициран начин за „прихващане“ на стандартизирани грешки като резултат от извикване на методи.
- Примери:
  - Опит за делене на нула (java.lang.ArithmeticException)
  - Опит за отваряне на несъществуващ файл (java.io.FileNotFoundException)
  - Грешка при вход/изход (java.io.IOException)

Прихващане на изключения

```
try{
   <програмен код>
}
catch (<изключение> <име>){
   <програмен код, който се изпълнява при
изключение>
}
finally{
   <програмен код, който се изпълнява винаги,
въпреки изключение>
}
```

Пример

```
 // Izchisliava x ot a.x + b = 0
int a = 0;
int b = 4;
try{
  System.out.println("x = "+(-b/a));
}
catch (java.lang.ArithmeticException e){
  if (b == 0) System.out.println("Vs. x e resh");
  else System.out.println("Niama rehesnie");
}
```

НО! Това е лош пример! По-добре е да се грижим ние сами за данните си!

Прихващане на повече от 1 изключение

```
java.io.PrintStream out = null;
try {
  out = new java.io.PrintStream(new java.io.FileOutputStream("out.txt"));
  int i = 123;
  out.print("Hello");
  out.println(" world "+i);
}
catch (java.io.FileNotFoundException e) {
       System.err.println("File Not Found");
}
catch (java.io.IOException e) {
       System.err.println("IO error has occured!");
}
finally {
   if (out!=null) out.close();
}
```

Изключения и „Stack trace“

- Най-общата група изключения е изключението „Exception“ (т.е. ако напишете catch (Exception e){…}).
- Допълнителна информация за изключението може да се получи от т.нар. „Stack trace“.

```
int a = 0;   int b = 4;
try{
  System.out.println("x = "+(-b/a));
}
catch (java.lang.ArithmeticException e){
  e.printStackTrace();
}
```

Хвърляне на изключения

- Възможно е писането на методи, които вместо да обработват сами изключенията на операторите вътре в тях, да прехвърлят изключенията на извикващите ги методи.
- Това е нещо като „прехвърляне на отговорността“.
- Такава функционалност се получава чрез оператора „throw“.

Пример

```
public static void showFileContents(String filename)
        throws java.io.FileNotFoundException,
java.io.IOException {
    java.io.BufferedReader in =
            new java.io.BufferedReader(
                    new java.io.FileReader(filename));
   int readbyte;
   while ((readbyte = in.read()) != -1) {
      System.out.print((char)readbyte);
   }
   in.close();
 } public static void main(String[] args) {
      try{  showFileContents("input.txt");    }
      catch (java.io.FileNotFoundException e){
        System.err.println("File not found");
      } catch (java.io.IOException e){
        System.err.println("IO error has occured"); }}
```

Още за грешките

- Каскадно обработване на грешки
  - Виж примера в сайта
- Ръчно „хвърляне“ на грешка

```
if (divisor == 0) throw new
java.lang.ArithmeticException("You divide by 0");
```

```
. . .  System.err.println(e.getMessage());
```

„лошия пример“

```
// Izchisliava x ot a.x + b = 0
double a = 0; double b = 4;
try{
  if (a == 0){
    if (b == 0)
   throw new java.lang.ArithmeticException("Vsiako x e reshenie");
    else throw new java.lang.ArithmeticException("Niama reshenie");
  }
  System.out.println("x = "+(-b/a));
}
catch (java.lang.ArithmeticException e){
  System.out.println(e.getMessage());
}
```

Обект String

- Символните низове (String) съдържат поредица от символи от Unicode таблицата.
- един String наподобява масив от тип char[].
- към определен елемент (буква) от String трябва да се обръщаме чрез метод

```
String s = "Hello World";
System.out.println(s.charAt(6));
s.length();
```

Обект String

- С = не се копират:

```
String s = "Hello World";
String s2 = s;
```

- Така и двете променливи s и s2 сочат към един и същи символен низ.
- Непроменими („garbage collector“):

```
String s = "Hello World";
s = "abcd";
```

Операции с низове

- **1. Сравнение за еднаквост**: методът equals() връща true или false в зависимост дали подадения низ е същия `s1.equals(s2)`
- **2. Сравнение за еднаквост без взимане под внимание на малки и големи букви**: `s1.equalsIgnoreCase(s2)`
- **3. Лексикографска подредба**: compareTo() , compareToIgnoreCase()
- **4. Конкатенация на низове**: +, concat()
- **5. Извличане на част от низ:** substring(<индекс>, <индекс>)

Операции с низове

- **6. Търсене в символен низ:** indexOf() , lastIndexOf()
- **7. Промяна на подниз:** replace()
- **8. Превръщане на малки в големи букви и обратно:** toLowerCase(), toUpperCase()
- **9. Премахване на празни символи:** trim()
- **10. Превръщане на низ в число и обратно:** Integer.parseInt(num);

Динамични низове (StringBuffer)

- Недостатъците породени от непроменимост на низовете от тип String, могат да бъдат преодоляни чрез обект наречен StringBuffer.
- Обекти от този тип са вече истински динамични масиви от символи и поради тази причина можем директно да ги променяме без да има нужда от копиране на целия обект в нов буфер.

```
StringBuffer strb = new StringBuffer();
strb.append("Hello");
strb.append(" World!");
System.out.println(strb);
```

Динамични низове (StringBuffer)

- Задаване на капацитет

```
StringBuffer strb2 = new StringBuffer(200);
```

- length() и capacity().
  Length() връща броят символи, които са записани в StringBuffer, а capacity() връща общата му дължина (включително празните символи, които сме резервирали).
- setLength()
- insert(<индекс>, <низ>)
- setCharAt(<индекс>, <символ>)

Динамични низове (StringBuffer)

- delete(индекс, индекс) – изтриване на подниз
- reverse() – обръщане в обратен ред
- substring(), indexOf() и lastIndexOf()
- toString() – преобразуване на StringBuffer в String

- **Важно**: StringBuffer има възможност за синхронизация на нишки!
- StringBuilder – няма.

Регулярни изрази

Регулярен израз означава търсене на текст по даден шаблон

```
// Низ, който ще сравняваме
String email = "philip@abv.bg";
// Саздаваме шаблон
java.util.regex.Pattern p = null;
p = java.util.regex.Pattern.compile("^\\w+[@][a-zA-Z0-9\\-
]+[.][a-zA-Z]+$");
// Създаваме "търсач" чрез шаблон и низ
java.util.regex.Matcher m = p.matcher(email);

if (m.matches()) System.out.println(email+" looks like an e-
mail");
else System.out.println(email+" do not look like an e-mail");
```

Регулярни изрази

- text – търси думата „text“;
- [a-z] – малки букви;
- [^a-z] – всички символи БЕЗ малките букви;
- [A-Z] – главни букви;
- [^A-Z] – всички символи БЕЗ главните букви;
- [0-9] – цифри от 0 до 9;
- [^0-9] – всички символи БЕЗ цифри;
- [a-zA-Z] – малки и големи букви;
- [0-9a-zA-Z] – малки и големи букви и цифри;
- [^%ABW] – всички символи без % и буквите A, B и W;
- [a-z-[mnp]] – малки букви от a до z, но без буквите m, n и p;
- [a-z-[m-p]] – малки букви от a до z, но без буквите от m до p;
- \s – интервал, табулация, нов ред, връщане в начало на ред (еквивалентно на [\t\n\r]);
- \S – всички символи БЕЗ интервал, табулация, нов ред и начало на ред ([^ \t\n\r]);
- \d – цифра (еквивалентно на [0-9]);
- \D – всички символи БЕЗ цифрите (еквивалентно на [^0-9];
- \w – букви, цифри и долна черта (еквивалентно на [a-zA-Z0-9_]);
- \W – всички символи БЕЗ букви, цифри и долна черта ([^a-zA-Z0-9_]).

Регулярни изрази

Операции

- + – „един или повече“, например „[a-z]+“ означава „търсим една или повече малки букви;
- * – „нула или повече“, например „[A-Z]*“ означава „нула или повече главни букви“;
- ? – „нула или една“, например „[0-9]?“ означава „нула или една цифра“.

- ^ – „започва с…“, например „^\w+“ означава „започва с една или повече букви, цифри или долна черта;
- $ – „завършва с…“, например „[0-9]$“ означава „завършва с цифра“.

Домашни задания №1

- Варианти за работа в екип от 2-ма души
- Самостоятелни задачи

- Предаване и защита 3-та седмица!