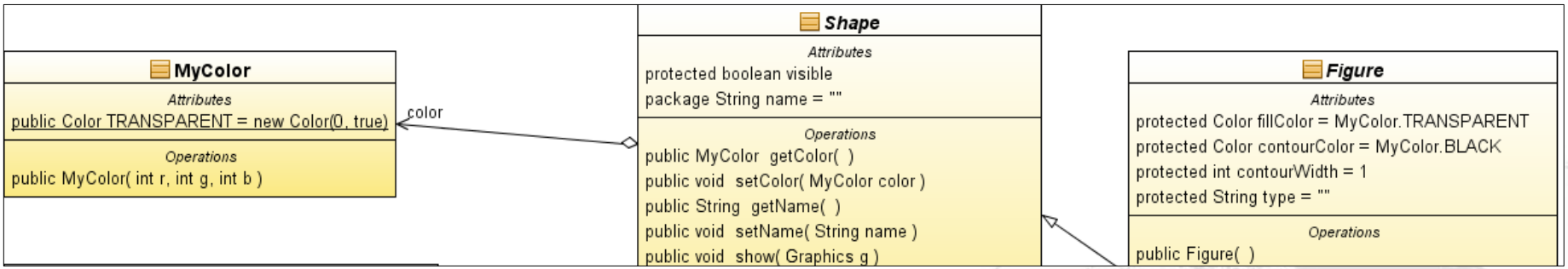
Trayan Iliev
IPT – Intellectual Products & Technologies Ltd.

Основни типове данни и езикови конструкции в езика Java™. Спецификация на изискванията.



Траян Илиев

IPT – Intellectual Products & Technologies
e-mail: tiliev@iproduct.org
web: http://www.iproduct.org

Съдържание

1. Основни елементи на езика Java™
2. Типове данни
3. Създаване на нови типове – класове, методи и атрибути
4. Променливи и константи
5. Математически, логически, релационни, побитови и низови оператори
6. Преобразуване на типове
7. Основни езикови конструкции за управление хода на програмата
   if-else, do-while, for, break, continue, switch
8. Низове и регулярни изрази в Java™
9. Коментари и стил на документиране.
10. Създаване, компилиране и изпълнение на Java™ програма
11. Спецификация на изискванията към софтуера – XP/UML

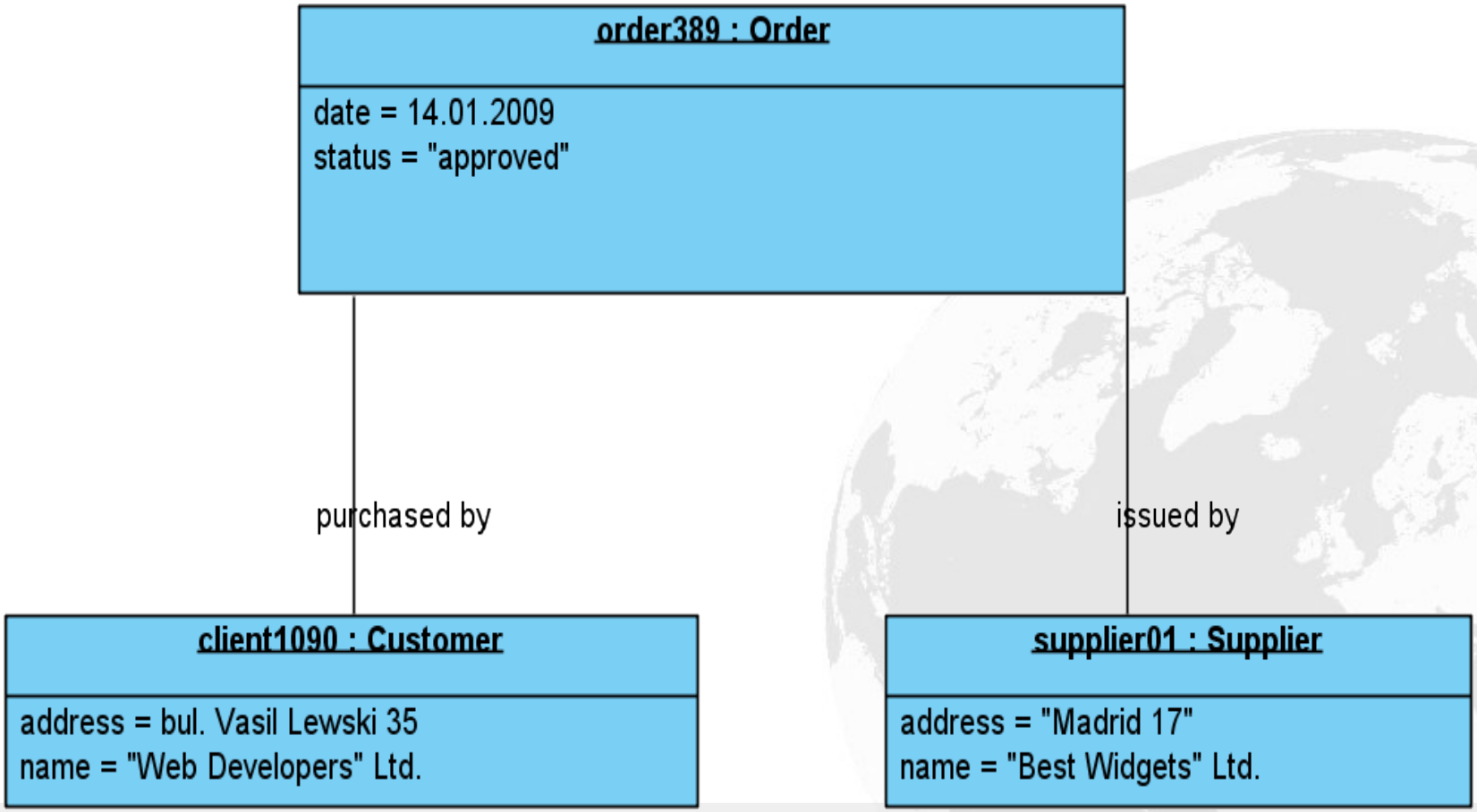
Основни елементи на езика Java™ - типове данни, променливи и константи

- Обекти и референции
- Създаване на обекти
- Примитивни и обектни типове данни
- Структури от данни - масиви
- Полета и методи на обект
- Използване на готови библиотеки
- Статични атрибути и методи - static
- Променливи и константи - final

Обекти и референции

- Клас – множество от обекти, които споделят обща структура, поведение и възможни връзки с обекти от други класове = тип на обектите
  - структура = атрибути, свойства, член променливи
  - поведение = методи, операции, член функции, съобщения
  - връзки с обекти от други класове: асоциация, агрегация, композиция – моделират се като атрибути на класа – референции към обекти от свързания клас
- Обектите се явяват инстанции на класа, който имат:
  1) собствено състояние
  2) уникален идентификатор = референция към обекта

Диаграма на обекти



Създаване на обекти

- Клас String – моделира низ от символи:
  - декларация:

    String s;
  - инициализация (на отделен ред):

    s = new String(“Hello Java World”);
  - декларация + инициализация:

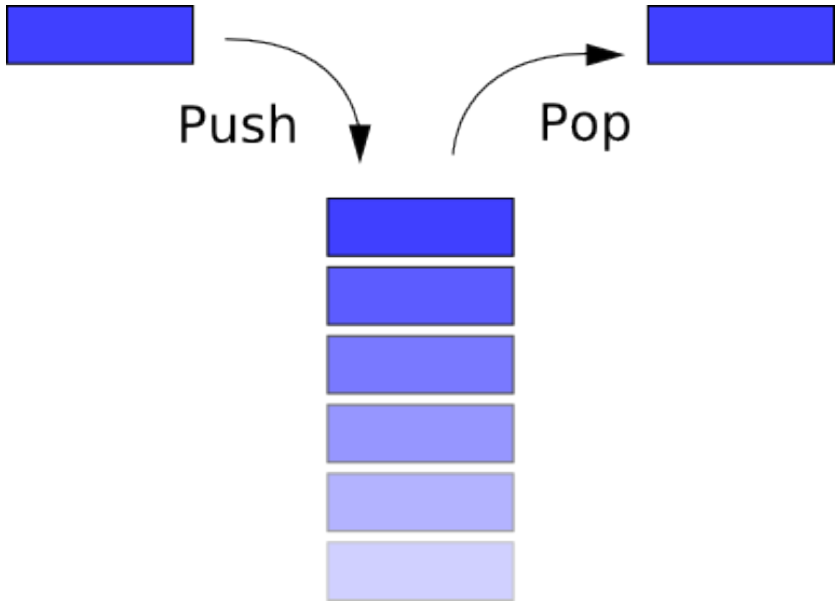
    String s = new String(“Hello Java World”);
  - декларация + инициализация (по-кратка форма, важи само за класа String):

    String s = “Hello Java World”;

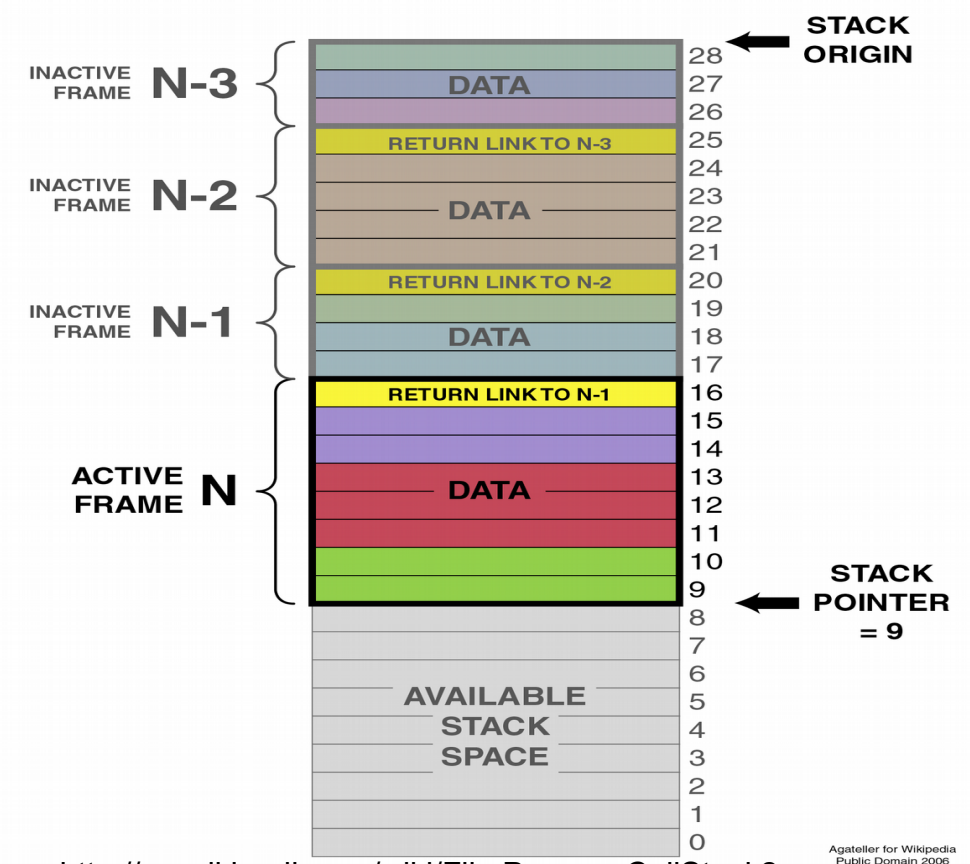
Видове памет

- Регистрова памет – регистри на процесора, бързи, малък брой, съхраняват операндите на инструкциите точно преди обработка
- Програмен стек = Last In, First Out (LIFO) – съхранява примитивните типове данни и референции към обектите по време на изпълнение на програмата
- Динамично алокируема памет – Heap – може да съхранява различни по големина обекти за различен период от време, могат да се създават нови обекти динамично, както и да се освобождават – Garbage Collector
  - Young generation – обекти които съществуват за кратко
  - Old generation – обекти съществуващи по-дълго
  - ~~Permanent Generation = class definitions.~~ → Java 8 Metaspace
- Статична памет, постоянна памет, външна памет

Програмен стек (Wikipedia)



http://en.wikipedia.org/wiki/File:Data_stack.svg



http://en.wikipedia.org/wiki/File:ProgramCallStack2.png

Примитивни и обектни типове данни

- Примитивни типове данни, обектни обвиващи типове и стойности по подразбиране за атрибути от примитивен тип
  - boolean --> Boolean — false
  - char --> Character — '\u0000'
  - byte --> Byte — (byte) 0
  - short --> Short — (short) 0
  - int --> Integer — 0
  - long --> Long — 0L
  - float --> Float — 0.0F
  - double --> Double — 0.0D
  - void --> Void

Обектни типове данни

- Създаване на клас (нов тип данни)

```
class MyClass { /* атрибути и методи на класа */ }
```

- Създаване на обект от класа MyClass :

```
MyClass myObject = new MyClass();
```

- Деклариране и инициализация на атрибути:

```
class Person {
    String name;
    int age;
}
```

- Достъп до атрибут: Person p1 = new Person();

```
p1.name = “Ivan Petrov”;      p1.age = 28;
```

Обектни типове данни

- Инициализация със стойности по подразбиране
- Стойност на неициализирана референция = **null**
- Деклариране на методи на класа

```
class Person {
    String name;
    int age;
    String setNameAndAge (String aName, int anAge) {
        name = aName;
        age = anAge;
        return “Name: “ + name + “Age: “ + age;
    }
}
```

Име на метода

Аргументи

Тип на връщане

Тяло на метода

Връщане на стойност

Коментари и стил на документиране

- Коментари:
  - едноредови //,
  - на няколко реда /* ... */
- JavaDoc стандарт за описание на вградена в кода документация – HTML, тагове: @author, {@code}, {@docRoot}, @deprecated, @exception, {@inheritDoc}, {@link}, {@linkplain}, {@literal}, @param, @return, @see, @serial, @serialData, @serialField, @since, @throws, {@value}, @version
- Автоматично генериране на JavaDoc документация в HTML формат
- Оформление на кода

Литерали от примитивен тип

- в десетична бройна система:
  int: 145, 2147483647, -2147483648
  long: 145L, -1l, 9223372036854775807L
  float: 145F, -1f, 42E-12F, 42e12f
  double: 145D, -1d, 42E-12D, 42e12d
- в шестнайсетична бройна система:
  0x7ff, 0x7FF, 0X7ff, 0X7FF
- в осмична бройна система: 0177
- в двоична бройна система: 0b11100101, 0B11100101

Операторите в Java™ - I

- Оператор за присвояване
- Математически оператори
- Релационни оператори
- Логически оператори
- Побитови оператори
- Низови оператори
- Оператори за преобразуване на типовете
- Приоритети на операторите

Операторите в Java™ - II

- Всеки оператор има приоритет и асоциативност – например + и – имат по-нисък приоритет от * и /
- Приоритетът може да се зададе явно с помощта на скоби (и) - например (y – 1) / (2 + x)
- Според асоциативността операторите биват ляво-асоциативни, дясно-асоциативни и не-асоциативни:
  Например: x + y + z => (x + y) + z, защото операторът + е ляво-асоциативен;

  ако беше дясно асоциативен, резултатът би бил x + (y + z)

Операторите в Java™ - III

- Оператор за присвояване: **=**
  - не е симетричен – т.е. **x = 42** е OK, **42 = x** НЕ е
  - отляво винаги стои променлива от определен тип а отдясно израз от същия тип или тип, който може да бъде автоматично преобразуван до дадения
- Математически оператори:
  - с един аргумент (унарни): **-, ++, --**
  - с два аргумента (бинарни): **+, -, *, /, %** (остатък)
- Комбинирани: **+=, -=, *=, /=, %=**
  Например: **a += 2** <=> **a = a + 2**

Предаване на аргумент по референция и по стойност - I

- Формални и фактически аргументи – Пример:

Статичен меттод – няма **this**

Формален аргумент
– копира стойността на фактическия

```
public static void incrementAgeBy10(Person p){
    p.age = p.age + 10;
}
Person p2 = new Person(23434345435L, "Petar
Georgiev", "Plovdiv", 39);
incrementAgeBy10(p2);
System.out.println(p2);
```

Фактически аргумент

Предаване на аргумент по референция и по стойност - II

- Случай А: Когато аргументът е от **примитивен тип** формалният аргумент **копира стойността** на фактическия
- Случай В: Когато аргументът е от **обектен тип** формалният аргумент **копира референцията** сочена от фактическия
- В случаи А и В: Промените в копието (формалния аргумент) **не се отразяват** на фактическия
- Ако обаче формалният и фактическият аргумент сочат към един и същи обект (Случай В) – то **промените в свойствата (стойностите на атрибутите) на този обект стават достъпни от извикващия метод** – т.е. можем да върнем стойност чрез този аргумент.

Операторите в Java™ - IV

- Релационни оператори (сравнение): **==, !=, <=, >=**
- Логически оператори: **&&** (AND), **||** (OR) и **!** (NOT)
  - изразът се изчислява отляво надясно само докато е необходимо за определяне на крайния резултат
- Побитови оператори: **&** (AND), **|** (OR) и **~** (NOT), **^** (XOR), **&=, |=, ^=,**
  - побитово изместване: **<<**, **>>** (запазва знака), **>>>** (вмъква винаги нули отляво - не запазва знака), **<<=, >>=, >>>=**

Операторите в Java™ - V

- Троен **if-then-else** оператор:

  <boolean-expr> ? <then-value> : <else-value>
- Низов оператор за конкатенация: **+**
- Оператори за явно преобразуване на типовете:

  (byte), **(short)**, **(char)**, **(int)**, **(long)**, **(float)** ...
- Приоритети на операторите:

  унарни > бинарни аритметични > релационни > логически > три-аргументен оператор **if-then-else** > оператори за присвояване на стойност

Конструкции за управление хода на програмата в езика Java™ - I

- Условен оператор - **if-else**
- Връщане на стойност – **return**
- Оператори за организиране на цикъл - **while, do while, for, break, continue**
- Оператор за избор на една от много възможности - **switch**

Конструкции за управление хода на програмата в езика Java™ - II

- Условен оператор **if-else**:

if(<boolean-expr>)
** <then-statement>**

или

if(<boolean-expr>)
** <then-statement>**
else
** <else-statement>**

Конструкции за управление хода на програмата в езика Java™ - III

- Връщане на стойност с излизане от метода:
  return; или **return <value>;**
- Оператор за организиране на цикъл **while**:

```
while(<boolean-expr>)
    <body-statement>
```

- Оператор за организиране на цикъл **do-while**:

```
do <body-statement>
while(<boolean-expr>);
```

Конструкции за управление хода на програмата в езика Java™ - IV

- Оператор за организиране на цикъл **for**:

```
for(<initialization>; <boolean-expr>; <step>)
    <body-statement>
```

- Оператор за организиране на цикъл **foreach**:

```
for(<value-type> x : <collection-of-values>)
    <body-statement-using-x>
```

Пример:

```
for(Point p : pointsArray)
    System.out.println("(" +p.x + ", " + p.y + ")");
```

Конструкции за управление хода на програмата в езика Java™ - V

- Оператори за излизане от блок (цикъл) **break** и за излизане от итерация на цикъл **continue**:

```
<loop-iteration> {
   //do some work
   continue;  // goes directly to next loop iteration
   //do more work
   break;  // leaves the loop
   //do more work
}
```

Конструкции за управление хода на програмата в езика Java™ - VI

- Използване на етикети с **break** и **continue**:

```
outer_label:
<outer-loop> {
    <inner-loop> {
        //do some work
        continue; // continues inner-loop
        //do more work
        break outer_label; // breaks outer-loop
        //do more work
        continue outer_label;  // continues outer-loop
}
```

Конструкции за управление хода на програмата в езика Java™ - VII

- Избор на една от няколко възможности **switch**:

```
switch(<selector-expr>) {
  case <value1> : <statement1>; break;
  case <value2> : <statement2>; break;
  case <value3> : <statement3>; break;
  case <value4> : <statement4>; break;
  // more cases here ...
  default: <default-statement>;
}
```

Низове

- Класът **String** предоставя **immutable** обекти – т.е. всяка операция върху низа създава нов обект в хипа
- **StringBulider** – предоставя ефикасен откъм ресурси начин да модифициране на низове, като реализира Reusable Design Pattern: **Builder** – за постъпково изграждане на низа (основно с метод **append** и **insert**)
- Основни операции в класа **String**. Форматиран изход – метод **format()** и клас **Formatter**. Спецификатори:

%[argument_index$][flags][width][.precision]conversion

Конверсия на типа при форматиране

- d – decimal, интегрални типове
- c – character (unicode)
- b - boolean
- s - String
- f – float, double (с десетична точка)
- e - float, double (scientific notation)
- x – шестнайсетична стойност на интегрални типове
- h – шестнайсетичен хеш код

Регулярни изрази (1)

- Символни класове
  - **.** Any character (may or may not match line terminators)
  - **\d** A digit: [0-9]
  - **\D** A non-digit: [^0-9]
  - **\s** A whitespace character: [\t\n\x0B\f\r]
  - **\S** A non-whitespace character: [^\s]
  - **\w** A word character: [a-zA-Z_0-9]
  - **\W** A non-word character: [^\w]

Регулярни изрази (2)

- Квалификатори:
  - **X?** X, once or not at all
  - **X*** X, zero or more times
  - **X+** X, one or more times
  - **X{n}** X, exactly n times
  - **X{n,}** X, at least n times
  - **X{n,m}** X, at least n but not more than m times
- **Greedy, Reluctant (?) & Possessive (+)** квалификатори
- **Capturing Group - (X)**

Регулярни изрази (3)

- Клас **Pattern** – основни методи:
  - **public static Pattern compile(String regex)**
  - **public Matcher matcher(CharSequence input)**
  - **public static boolean matches(String regex, CharSequence input)**
  - **public String[] split(CharSequence input, int limit)**
- Клас **Matcher** – основни методи:
  - **public boolean matches()**
  - **public boolean lookingAt()**
  - **public boolean find(int start)**
  - **public int groupCount()** и **public String group(int group)**

Литература и интернет ресурси

- Екел, Б., Да мислим на JAVA. Софтпрес, 2001.
- Oracle® Java™ Technologies webpage – http://www.oracle.com/technetwork/java/index.html
- How to Write Doc Comments for the Javadoc Tool – http://java.sun.com/j2se/javadoc/writingdoccomments/index.html
- javadoc - The Java API Documentation Generator – http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/javadoc.html#doclets

Благодаря Ви за вниманието!

Въпроси?