# ЛЕКСИЧНІ ОСНОВИ МОВИ ПРОГРАМУВАННЯ JAVA

Питання 1.3.

# Змінні

- Приклад оголошення змінної
  - double salary;
  - int vacationDays;
  - long earthPopulation;
  - char yesChar;
  - boolean done;
- Ім’я змінної повинно починатись з букви та складатись з букв та цифр
  - Буквами вважаються символи ‘A’ – ‘Z’, ‘_’ та будь-який юнікод-символ, що відповідає букві (π, ä та інші)
  - Аналогічно, цифрами вважаються як десяткові, так і цифри з конкретної мови
  - Не можна використовувати пробіл, ‘+’, ‘©’ та інші
  - Регістр символу в назві теж враховується, довжина назви не обмежується
- Визначити, які символи Java вважає буквами, можна за допомогою методів isJavaIdentifierStart() та isJavaIdentifierPart() з класу Character.
  - Хоч знак $ вважається буквою Java, для іменування елементів його використовувати не рекомендується.
  - Він слугує для позначення назву, які формуються компілятором та інструментальними засобами.

- В якості назв змінних не можна використовувати зарезервовані слова Java.

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

- В одному рядку можна оголосити кілька змінних, хоч такий підхід і не рекомендується:
  - int i, j;

# Ініціалізація змінних

- Виконується за допомогою оператора присвоєння

```
int vacationDays;
System.out.println(vacationDays); // ОШИБКА! Переменная не инициализирована
```

```
double salary = 65000.0;
System.out.println(salary);
int vacationDays = 12; // здесь можно объявить переменную
```

- Рекомендується оголошувати змінну якомога ближче до місця її використання.

# Константи

- Для позначення констант використовується ключове слово final
  - Присвоїти значення такій змінній можна лише 1 раз.
  - Рекомендується в іменах констант використовувати лише великі букви

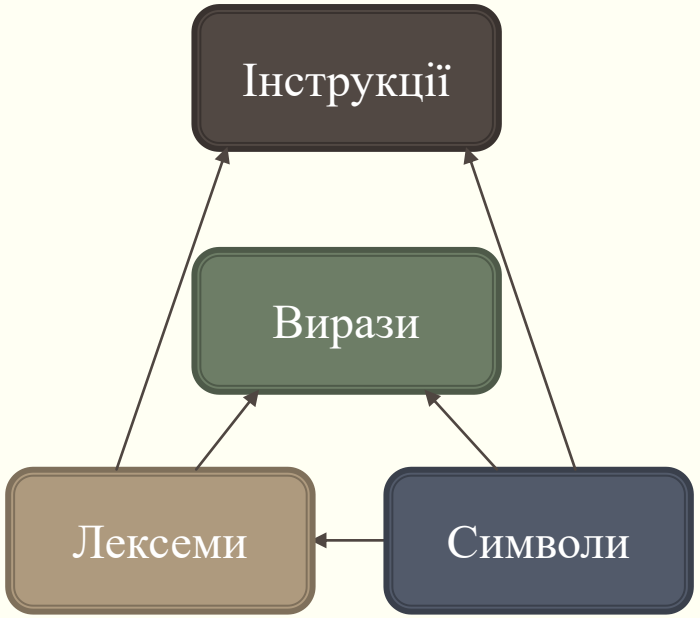
```
public class Constants
{
  public static void main(String[] args)
  {
    final double CM_PER_INCH = 2.54;
    double paperWidth = 8.5;
    double PaperHeight = 11;
    System.out.println("Paper size in centimeters: "
      + paperWidth * CM_PER_INCH + "by" + paperheight * CM_PER_INCH);
  }
}
```

# Статичні константи

- Часто константа повинна бути видимою декільком методам з класу.
  - Такі константи називають константами класу і задаються за межами методів
  - Використовуються ключові слова static final, а також public – для отримування доступу

```
public class Constants2
{
  public static final double CM_PER_INCH = 2.54;
  public static void main(String[] args)
  {
    double paperWidth = 8.5;
    double paperHeight = 11;
    System.out.println("Paper size in centimeters: "
      + paperWidth * CM_PER_INCH + " by " + paperHeight * CM_PER_INCH);
  }
}
```

# Вирази (expressions)




- Для ініціалізації змінних та інших цілей використовуються вирази – комбінація літералів, назв змінних, викликів методів та операторів.
  - Під час виконання вираз отримує значення, яке відповідає його типу даних.
  - Якщо вираз присвоєно змінній, типи змінної та виразу мають узгоджуватись; інакше компілятор повідомить про помилку.
  - Java розпізнає прості та складені (compound) вирази.

# Прості вирази

- Простим виразом є літерал (постійне значення, що присвоюється), назва змінної (містить значення) або виклик методу (повертає значення).
- Типи літералів у Java: string, булеві true і false, символ, ціле число, дробове число, null
- Виклик методу, який не повертає значення (void method) є спеціальним випадком простого виразу, наприклад,

```
System.out.println("Hello, World!");
```

- Такий вираз не можна присвоїти змінній
  - Спроба це зробити (наприклад, int i = System.out.println("X");) спричиняє повідомлення про помилку від компілятора.

# Літерали

- Рядковий літерал – послідовність Unicode-символів, що знаходяться між подвійними лапками, наприклад, "The quick brown fox jumps over the lazy dog".
  - Може також містити управляючі послідовності друкованих або недрукованих символів, що не можуть інакше бути присутніми у літералі.
  - Наприклад, символ “ використовує управляючу послідовність \”.
- Символьний літерал – Unicode-символ в одинарних лапках (‘A’).
- Булевий літерал складається із зарезервованого слова true або false.
- Цілочисельний літерал містить послідовність цифр.
  - До літералу типу long необхідно дописувати суфікс L.
- Основні формати представлення:
  - Десятковий (decimal)
  - Шістнадцятковий (hexadecimal) – з префіксом 0x або 0X (наприклад, 0x7B)
  - Вісімковий (octal), який починається з 0, наприклад, 0177.

# Літерали

```
public class SimpleExpressions
{
   public static void main(String[] args)
   {
      int counter = 10;
      double temperature = 98.6; // Assume Fahrenheit scale.
      String firstName = "Mark";
      int[] ages = { 52, 28, 93, 16 };
      char gradeLetters[] = { 'A', 'B', 'C', 'D', 'F' };
      float[][] matrix = { { 1.0F, 2.0F, 3.0F }, { 4.0F, 5.0F, 6.0F }};
      int x = 1, y[] = { 1, 2, 3 }, z = 3;
      double π = 3.14159;
      System.out.println(counter);
      System.out.println(temperature);
      System.out.println(ages.length);
      System.out.println(gradeLetters.length);
      System.out.println(matrix.length);
      System.out.println(x);
      System.out.println(y.length);
      System.out.println(z);
      System.out.println(p);
   }
}
```

Літерали з плаваючою крапкою мають цілочисельну та дробову частини, розділені крапкою, наприклад, 0.1.

- Можуть також бути присутні експонента (буква e або E) та суфікс типу (D, d, F або f), наприклад, 89F, 600D, 13.08E+23.

Літерал null присвоюється посилковій змінній (reference variable) і показує, що змінна не відноситься до об’єкту.

- Можна присвоїти змінній результат виконання методу:

```
boolean isLeap = isLeapYear(2012);
```

# Складені вирази (compound expressions)

- Послідовність простих виразів та операторів
- Оператор може бути
  - Унарним (наприклад, унарний -)
  - Бінарним (наприклад, +)
  - Тернарним (умовний оператор ?: )
- Оператори можна розділити на
  - Префіксні (унарні оператори перед операндом)
  - Постфіксні (унарні оператори після операнду)
  - інфіксні (бінарний або тернарний оператор, оточений операндами)

# Таблиця операторів

| Оператор | Символ | Опис |
|---|---|---|
| Додавання, конкатенація рядків | + | Операнд1 + Операнд2 |
| Віднімання, унарний мінус | - | Операнд1 - Операнд2, -Операнд3 |
| Індекс масиву | [] | Змінна[індекс] |
| Присвоєння | = | Змінна = Операнд |
| Побітовий І | & | Операнд1 & Операнд2 |
| Побітове доповнення | ~ | ~Операнд |
| Побітове виключне АБО | ^ | Операнд1 ^ Операнд2 |
| Побітове АБО | \| | Операнд1 \| Операнд2 |
| Зведення типу | (тип) | (тип) Операнд |
| Умовний оператор | ?: | Операнд 1 ? Операнд 2 : Операнд3 |
| Складене присвоєння | +=, -=, *=, /=, %=, &=, \|=, ^=, <<=, >>=, >>>= | Змінна *Оператор* Операнд |
| Остача від ділення | % | Операнд1 % Операнд2 |

# Таблиця операторів

| Оператор | Символ | Опис |
|---|---|---|
| Логічний І | && | Операнд1 && Операнд2 |
| Логічний АБО | \|\| | Операнд1 \|\| Операнд2 |
| Ділення | / | Операнд1 / Операнд2 |
| Рівність | == | Операнд1 == Операнд2 |
| Нерівність | != | Операнд1 != Операнд2 |
| Побітовий зсув вліво | << | Біти зліва зникають, а справа встановлюються 0 |
| Побітовий зсув вправо | >> | Біти справа зникають, а зліва встановлюються 0 |
| Логічне доповнення | ! | !Операнд |
| Логічне виключне АБО | ^ | Операнд1 ^ Операнд2 |
| Множення | * | Операнд1 * Операнд2 |
| Інкремент | ++ | змінна++ (постінкремент), ++змінна (преінкремент) |
| Декремент | -- | змінна-- (постдекремент), --змінна (предекремент) |
| Відношення порівняння | >,>=,<,<= | |

# Таблиця операторів

| Оператор | Символ | Опис |
|---|---|---|
| Оператор доступу | . | ідентифікатор1.ідентифікатор2 |
| Виклик методу | () | ідентифікатор(список аргументів) |
| Створення об'єкту | new | new ідентифікатор(список аргументів), виділяє пам'ять та викликає конструктор.<br>Для масивів може бути<br>new ідентифікатор[розмір масиву] |
| Перевірка типу відношення | instanceof | операнд1 instanceof операнд2, де операнд1 - об'єкт, а операнд2 – клас або інший користувацький тип даних |
| Беззнаковий побітовий зсув управо | >>> | Операнд1 >>> Операнд2, де кожен операнд повинен бути символом чи цілочисельним значенням |

# Коментарі

- Як і в решті мов програмування, ігноруються під час виконання.
  - Байткод для коментарів не будується.
- Використовуються для документування коду.
- У Java присутні три види коментарів:
  - Однорядкові коментарі (//)
  - Багаторядкові коментарі (/*...*/)
  - Коментарі для автоматичного форматування документації (javadoc)
- У Java коментарі /*...*/ не можуть бути вкладеними

# Коментарі

- Однорядкові коментарі корисні для вставки коротких, проте змістовних пояснень вихідного коду в цей же код.
  - `System.out.println(Math.sqrt(10 * 10 + 20 * 20));`

```
// Output distance from (0, 0) to (10, 20).
```

  - Не використовуйте для unhelpful documentation, типу

```
// Ця змінна зберігає цілочисельні значення.
```

- Багаторядкові коментарі:

```
/*

Рік вважається високосним, якщо він ділиться на 400

або ділиться на 4, але не ділиться на 100.

*/

System.out.println(year % 400 == 0 || (year % 4 == 0 && year % 100 != 0));
```

# Javadoc-коментарі

- Спрощує специфікацію зовнішньої HTML-based документації.
  - Спрощують підготовку технічної документації
- Це багаторядкові коментарі /** … */

```
/**
 *  Application entry point
 *
 *  @param args array of command-line arguments passed to this method
 */
public static void main(String[] args)
{
   // TODO code application logic here
}
```

- Між /** та */ знаходиться опис методу та Javadoc-тег @param (інструкція з префіксом @ для утиліти javadoc).

# Найбільш поширені теги

- @author ідентифікує автора вихідного коду.
- @deprecated визначає сутність у вихідному коді (наприклад, метод), що має більше не використовуватись.
- @param показує параметр методу.
- @see постачає see-also посилання.
- @since ідентифікує реліз, в якому сутність вперше створена.
- @return визначає тип (kind) значення, що повертає метод.
- @throws документує виключення, що викидається методом.

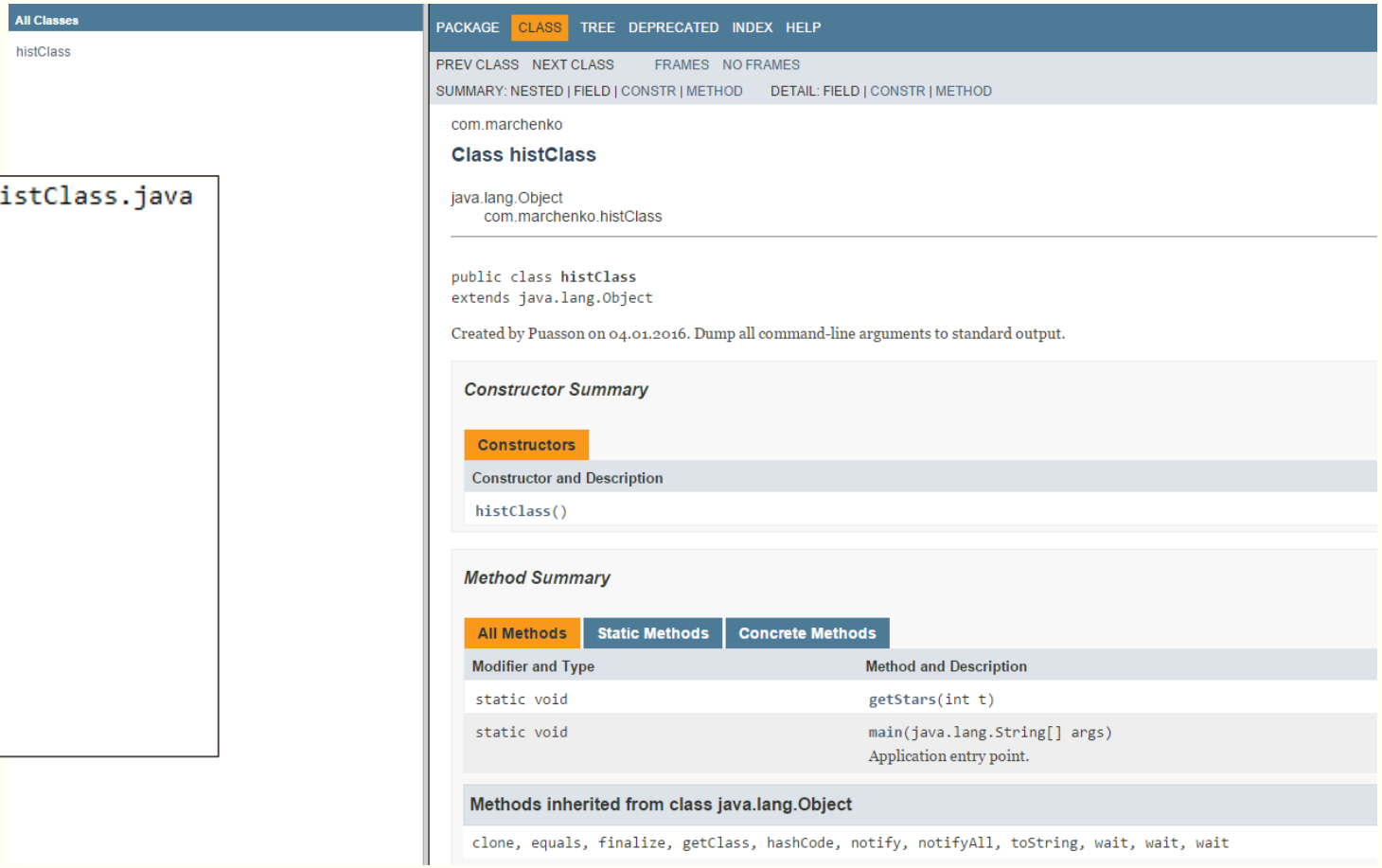
# Приклад генерації

- Задача про гістограму із зірочок

```
package com.marchenko;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.InterruptedIOException;

/**
  Created by Puasson on 04.01.2016.
  Dump all command-line arguments to standard output.
  @author Stanislav Marchenko
 */

public class histClass {
    /**
     Application entry point.
     @param args array of command-line arguments.
     @throws IOException for BufferReader
     * */
    public static void main(String[] args) throws IOException {
        System.out.println("Введіть виручку з першого магазину: ");
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        int t1 = Integer.parseInt(reader.readLine());// Double.parseDouble(reader.readLine());

        System.out.println("Введіть виручку з другого магазину: ");
        BufferedReader reader2 = new BufferedReader(new InputStreamReader(System.in));
        int t2 = Integer.parseInt(reader2.readLine());// Double.parseDouble(reader.readLine());

        System.out.println("Введіть виручку з третього магазину: ");
        BufferedReader reader3 = new BufferedReader(new InputStreamReader(System.in));
        int t3 = Integer.parseInt(reader3.readLine());// Double.parseDouble(reader.readLine());
```

# Згенерована документація

```
C:\Users\Puasson\IdeaProjects\hist\src\com\marchenko>javadoc histClass.java
Loading source file histClass.java...
Constructing Javadoc information...
Standard Doclet version 1.8.0_66
Building tree for all the packages and classes...
Generating .\com\marchenko\histClass.html...
Generating .\com\marchenko\package-frame.html...
Generating .\com\marchenko\package-summary.html...
Generating .\com\marchenko\package-tree.html...
Generating .\constant-values.html...
Building index for all the packages and classes...
Generating .\overview-tree.html...
Generating .\index-all.html...
Generating .\deprecated-list.html...
Building index for all classes...
Generating .\allclasses-frame.html...
Generating .\allclasses-noframe.html...
Generating .\index.html...
Generating .\help-doc.html...
```



# ДЯКУЮ ЗА УВАГУ!

**Наступне запитання:** управління ходом виконання Java-програми