Programare Avansata pe Obiecte
Laborator 1

Pahontu Bogdan-Ionut

E-mail: pahontubogdan@gmail.com

Contents

1. Understanding the Java Class Structure

1.1.Fields and methods

- Java classes have two primary **elements**:
  - **Methods:** operate on the state of the program
  - **Fields** (variables): hold the state of the program
- Together they are called the **members of the class.**
- The full declaration of a method is called a ***method signature***.

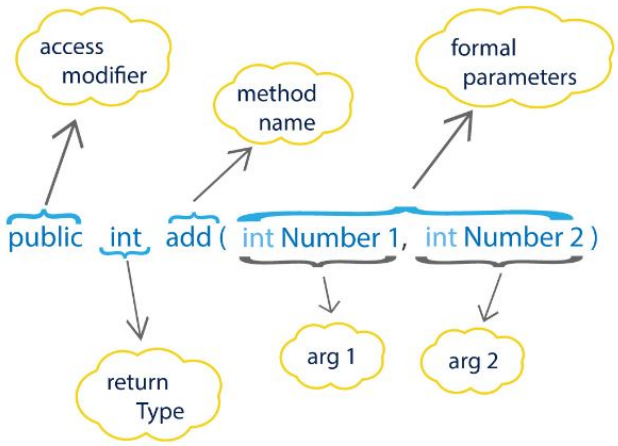


Figure 1 - Method Signature

Example:

```
package examples;

public class A_MethodSignature {
    public static void main(String[] args){
        System.out.println("Hello world !");
    }
}
```

Task 1:

1. Create packages lab1 → tasks →task1.

2. Create a class with two methods:
    a. Public static void main method;
    b. One method that is public and returns the sum of two integer numbers;

1.2.Ordering Elements in a class

Element	Example	Required?	Where does it go?
Package declaration	`package abc;`	No	First line in the file
Import statements	`import java.util.*;`	No	Immediately after the package
Class declaration	`public class C`	Yes	Immediately after the import
Field declarations	`int value;`	No	Anywhere inside a class
Method declarations	`void method()`	No	Anywhere inside a class

Figure 2 - Elements order

1.3.Comments

- Comments aren't executable code.
- Comments make your code easier to read.
- There are three types of comments in Java:
    - Single-line comment: // comment until end of line
    - Multiple-line comment:

/* Multiple

line comment

*/

    - Javadoc comment:

/**

* Javadoc multiple-line comment

* @author John

*/

1.4.Classes VS Files

- Most of the time, each Java class is defined in its own *.java file.
- You can even put two classes in the same file. When you do so, at most one of the classes in the file is allowed to be public.

```
public class C_PublicClass {
    public static void main(String args[]){
        System.out.println("This is the main method of the public class");
        AnotherClass.anotherMethod();
    }
}

class AnotherClass{
    public static void anotherMethod(){
        System.out.println("Hello from AnotherClass");
    }
}

/*public class ThirdClass{
     // This will generate a compiling error
}*/
```

- If you do have a public class, **it needs to match the file name**.
- Public class ThirdClass would not compile in a file named C_PublicClass.java.

1.5. JVM; JDK; JRE

- **JDK** – used for development and compiling the applications.
- Contains JRE and development tools:
    - Compiler (javac.exe) - converts java code into bytecode.
    - Java application launcher (java.exe) – loads the class and invokes its main method.

- **JRE** – used for running the application. Contains JVM, class libraries (like util, math,lang, awt, swing, etc) and other supporting files

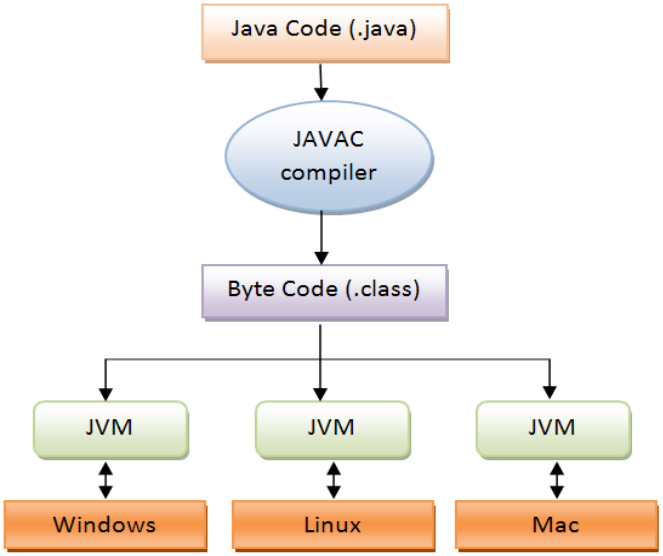


Figure 3 - JVM, JDK, JRE

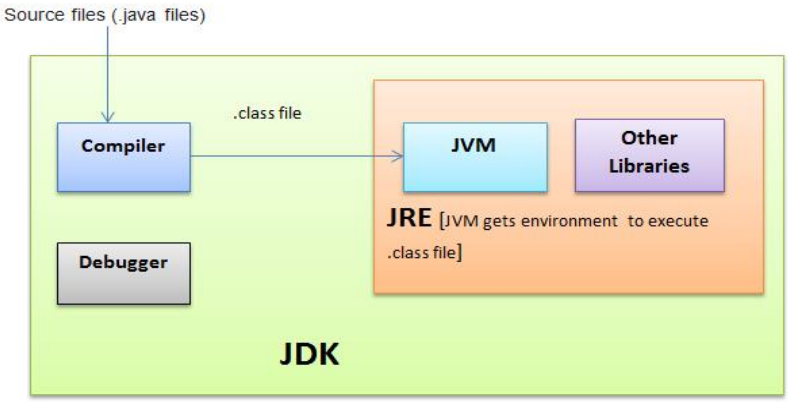


Figure 4 - JVM, JDK, JRE

2. Understanding Package (Declarations and Imports)

2.1.Importing packages

- In java we use packages to organize our classes and interfaces.

- Import tells the compiler which package to look in to find a class (ex. ImportExample class).
- **Wildcards: *** is the wildcard that matches all the classes in the package.
    - Imports only classes (not methods, child packages, fields)
- Redundant imports:
    - The java.lang package is automatically imported.
    - To import classes in the same package.
- We want to import the class named Path. This can be found in the package java.nio.file.Path.

  import java.nio.*; // NO GOOD – a wildcard only matches class names, not child packages.

  import java.nio.*.*; // NO GOOD – you can only have one wildcard and it must be at the end.

  import java.nio.file.Path.*; // NO GOOD – you cannot import methods, only class names.
- **Naming conflicts:** we use packages so that a class names doesn't have to be unique across all of Java.

  Example: Conflicts class (java.util.Date and java.sql.Date)
- **The first explicit import takes precedence over the second. (example: Conflicts class)**

  import java.util.Date;

  import java.sql.Date; --Error: *The import java.sql.Date collides with another import statement*.

Task 2:

1. Create packages lab1 → tasks →task2.
2. Create class Zoo with main method in task2 package.
3. Compile and run class Zoo.
4. Create class ZooArguments in task2 package.
5. Compile and run ZooArguments with 2 arguments, 1 argument and three arguments.

Obs:

javac package/*.java - compile everything in a package

In order to compile from command line please follow these commands:

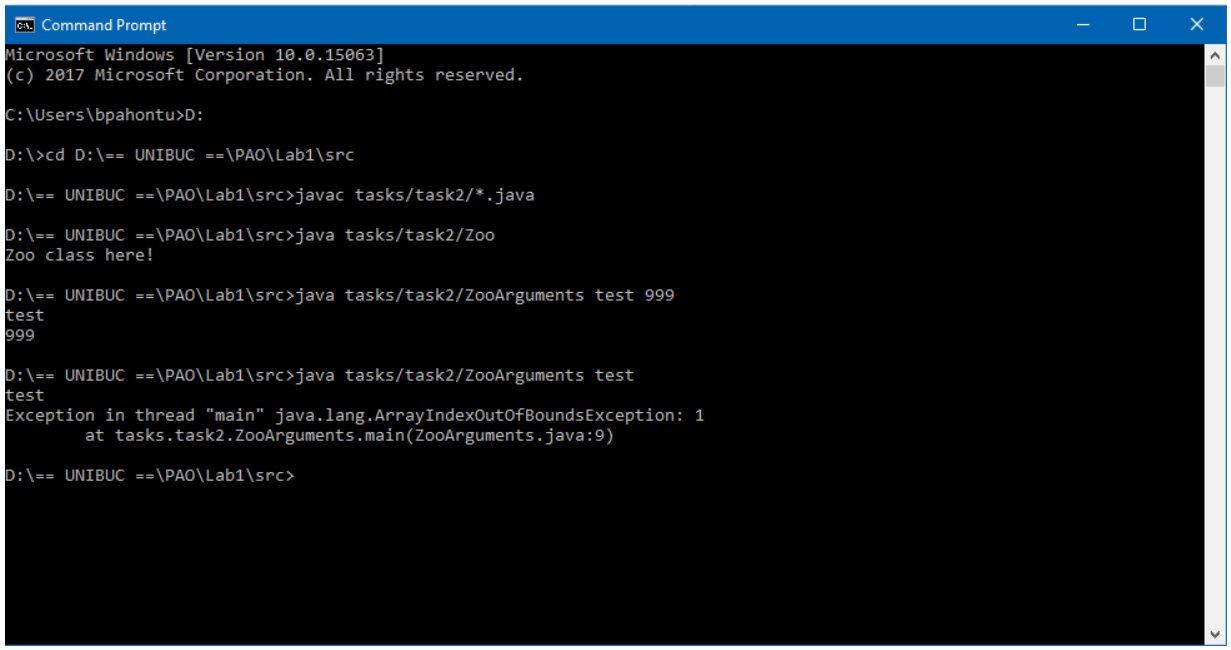


Figure 5 - Compile/Run program

Task 3:

1. Create package task3 into tasks package with two sub-packages: packagea and packageb
2. Each package will contain a Main class that will have one single public static void method named "**showPackage**". Method will return the package name.
3. Create a new package named mainpackage into tasks/ task3
4. Create a Main class with a public static void main function that calls showPackage from packagea and then showPackage from packageb

3. Object References and Primitives

3.1.Primitive types

Keyword	Type	Minimum Range	Maximum Range	Example
boolean	true or false	n/a	n/a	true
byte	8-bit integral value	-2^7 (-128)	2^7-1 (127)	123
short	16-bit integral value	-2^{15}	2^{15} - 1	123
int	32-bit integral value	-2^{31}	2^{31} - 1	123
long	64-bit integral value	-2^{63}	2^{63} - 1	123L
float	32-bit floating-point value	n/a	n/a	123.45f
double	64-bit floating-point value	n/a	n/a	123.456
char	16-bit Unicode value	n/a	n/a	'a'

Examples:

```
public class Primitives {
    //this will have the default value of a reference type
    public Object myObj;

    //int i = null; // compile time error -- can't set a null value to a primitive

    public static void main(String[] args) {

        //long max = 3123456789; // DOES NOT COMPILE
        long max = 3123456789L; // now Java knows it is a long
        System.out.println(max);

        // Since JAVA 7
        long creditCardNb = 1234_5678_9101_123L;

        F_PrimitivesVsReferences instance = new Primitives();
        System.out.println("What is value of myObjc : " + instance.myObj);

        String reference = "hello";
        int len = reference.length();
        System.out.println(len);
        //int bad = len.length(); // compile error -- there are no methods on
primitives

//        Integer itr = null; // this is ok
//        int j = itr; // this is also ok?
    }

}
```

Invalid uses of underscore:

```
float pi1 = 3_.1415F;       // Invalid; cannot put underscores adjacent to a decimal point
float pi2 = 3._1415F;       // Invalid; cannot put underscores adjacent to a decimal point
long socialSecurityNumber1
  = 999_99_9999_L;          // Invalid; cannot put underscores prior to an L suffix

int x1 = _52;               // This is an identifier, not a numeric literal
int x2 = 5_2;               // OK (decimal literal)
int x3 = 52_;               // Invalid; cannot put underscores at the end of a literal
int x4 = 5_______2;         // OK (decimal literal)

int x5 = 0_x52;             // Invalid; cannot put underscores in the 0x radix prefix
int x6 = 0x_52;             // Invalid; cannot put underscores at the beginning of a number
int x7 = 0x5_2;             // OK (hexadecimal literal)
int x8 = 0x52_;             // Invalid; cannot put underscores at the end of a number

int x9 = 0_52;              // OK (octal literal)
int x10 = 05_2;             // OK (octal literal)
int x11 = 052_;             // Invalid; cannot put underscores at the end of a number
```

Figure 6 - Invalid underscore using

3.2.Reference types

- A reference type refers to an object (instance of a class).
- References do not hold the value of the object they refer to.
- A reference “points” to an object by storing the memory address where the object is located.

java.util.Date today;

String greeting;

today = new java.util.Date();

- points to a new Date object in memory

greeting = "How are you?";

- points to a new String object.

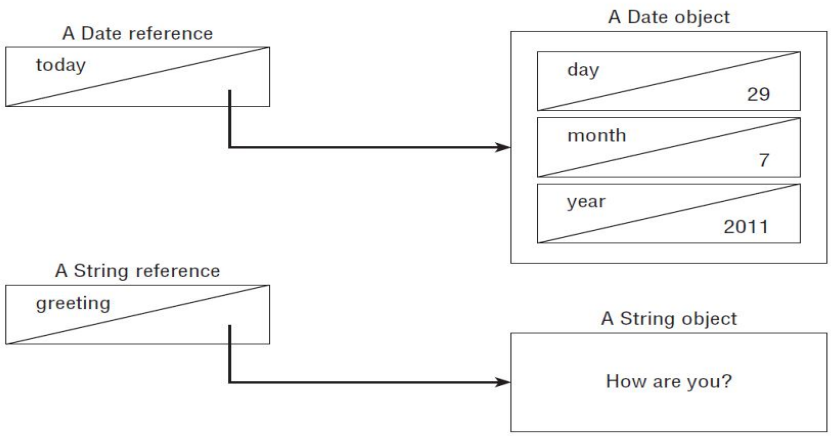


Figure 7 - Reference types

3.3. Differences between primitives and reference types

- Reference types can be assigned null and it's their default value.
- Primitive types will only give a compiler error if you attempt to assign them null

```
int value = null; // DOES NOT COMPILE
String s = null;
```

- Reference types can be used to call methods when they don't point to null.
- Primitives don't have methods declared on them.

4. Declaring and initializing variables

- We declare variables like this: String a; int b;
- Initializing means giving a value to the variable: a="test"; b=8;
- You can declare multiple variables in the same statement as long as they are all of the same type:

  String s3 = "yes", s4 = "no";

- How many are declared and initialized here?

  int i1, i2, i3 = 0;

int num, String value; // DOES NOT COMPILE

- Which of the following are legal (valid)?

  boolean b1, b2;

  String s1 = "1", s2;

  double d1, double d2;

  int i1; int i2;

  int i3; i4;

- Three rules to remember for legal identifiers:
  - The name must begin with a **letter** or the symbol **$** or **_**
  - Subsequent characters may also be numbers.
  - You cannot use the same name as a Java *reserved word*.
- Conventions: - methods and variables names begin with lowercase letter followed by CamelCase.
- Class names begin with uppercase letter followed by CamelCase.

5. Operators

Question:

1. What is the data types after performing the following operations:
   - int x = 1; long y = 33; → x*y
   - double x = 39.21; float y = 2.1f; → x+y
   - short x = 10; short y = 3; → x/y
   - short x = 14; float y = 13; double z = 30; → x*y/z

2. Response:
   - A. long.
   - B. Double

- C. Int
- D. First, x will automatically be promoted to int solely because it is a short and it is being used in an arithmetic binary operation. The promoted x value will then be automatically promoted to a float so that it can be multiplied with y. The result of x * y will then be automatically promoted to a double, so that it can be divided with z, resulting in a double value.

6. If-Then Statement

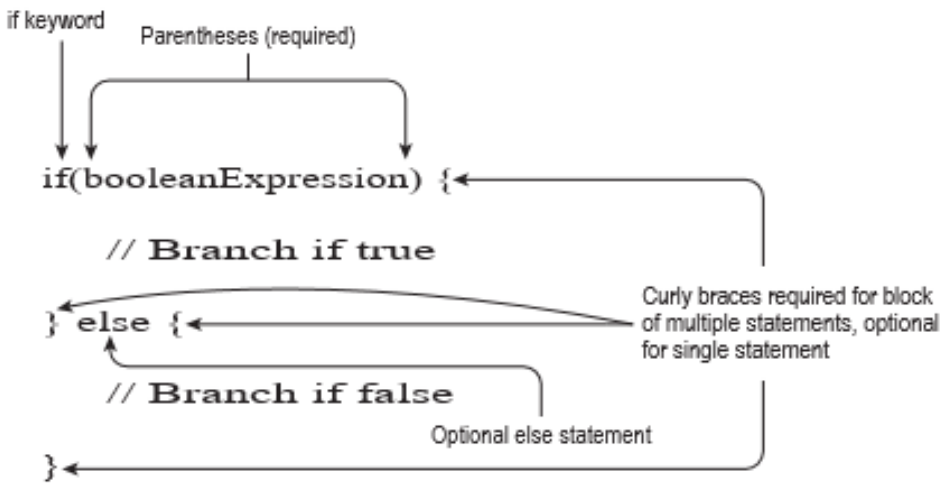


Figure 8 - If Statement

Ternary operator:

booleanExpression ? expression1 : expression2

7. Switch Statement

Switch can be performed on the following data types:

- int and Integer;
- byte and Byte;

- short and Short;
- char and Character;
- String;
- enums;

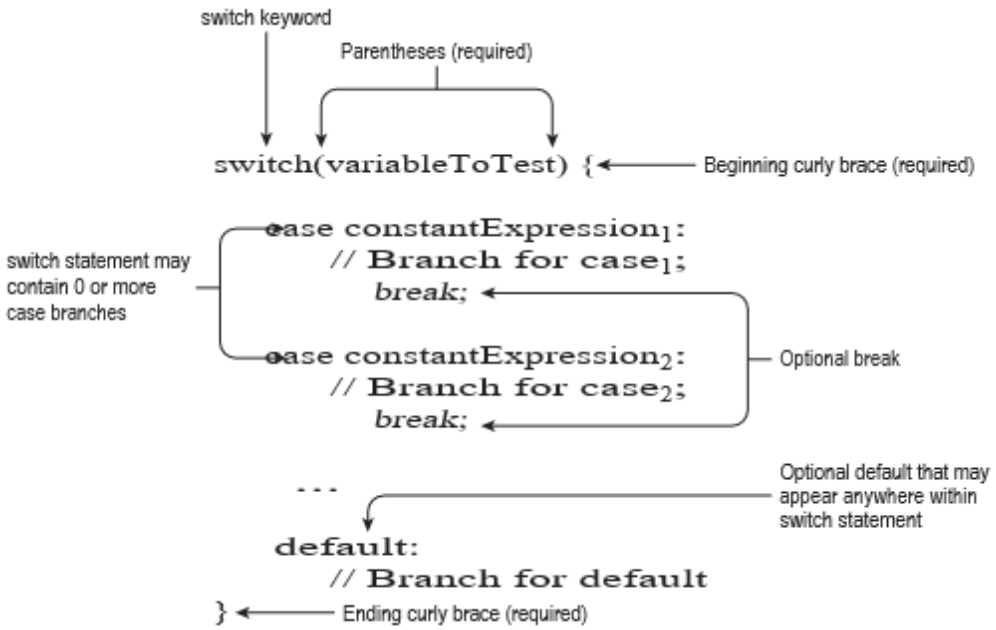


Figure 9 - Switch Statement

8. While / Do while

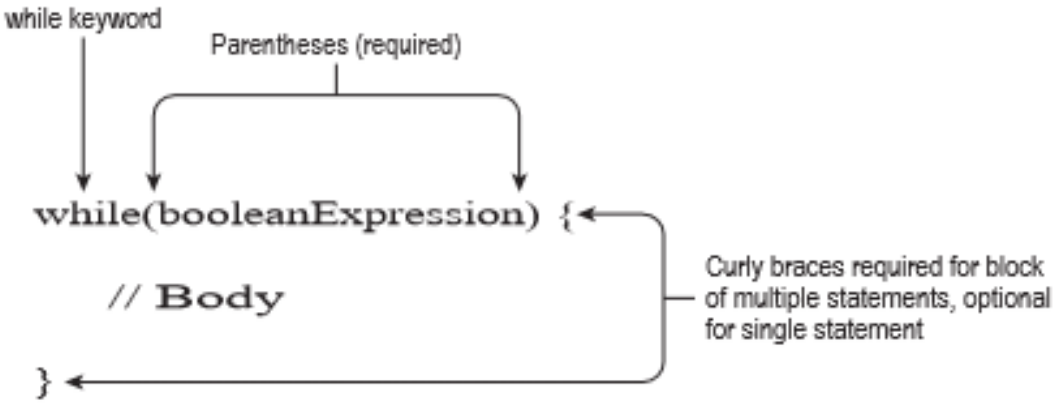


Figure 10 - While Statement

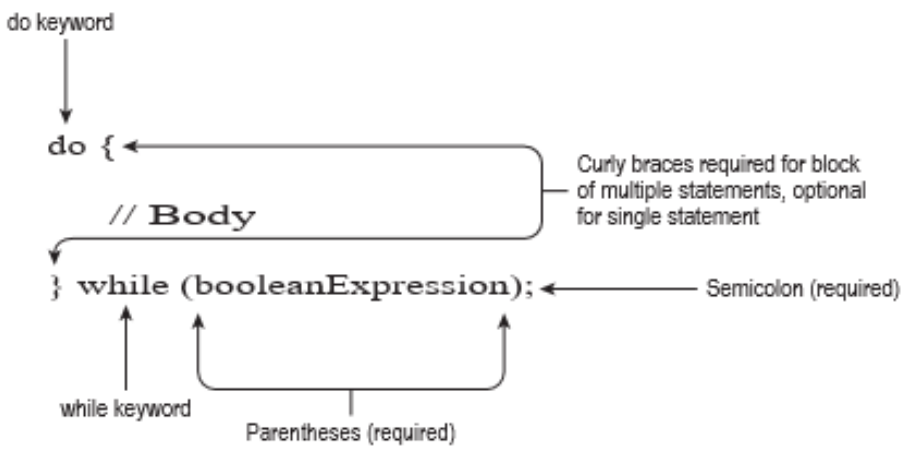


Figure 11 - DoWhile Statement

Task 4:

- Print the matrix:

  Input:

  int[][] matrix = {{5,2,1},{3,9,8},{5,7,3}};

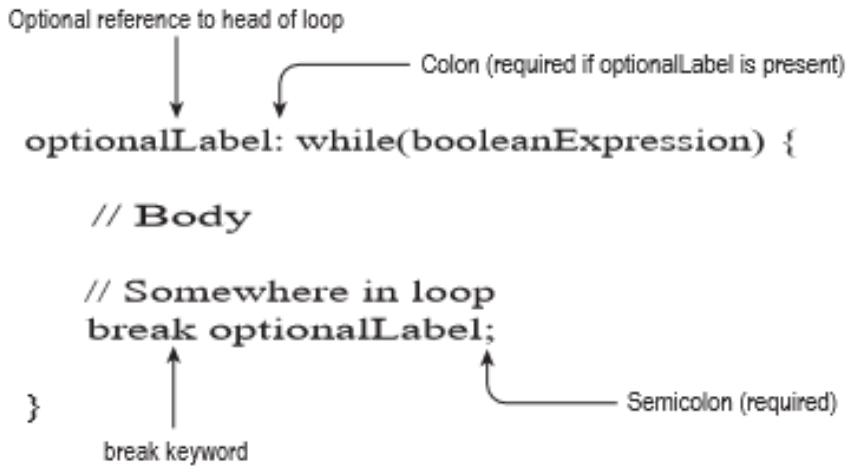
  OUTPUT:

  521

  398

  573

9. Break / Continue



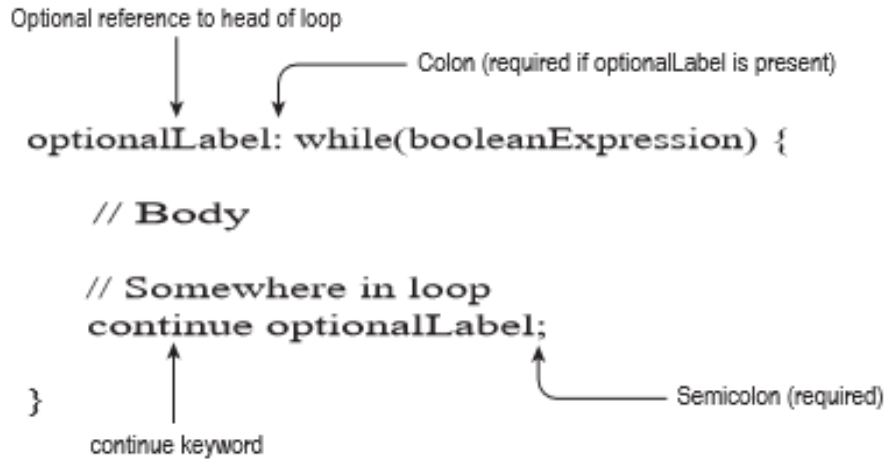


Figure 12 - Break & Continue

10.Scanner Class

Scanner is a class in java.util package used for obtaining the input.

Eample:

Task 5:

1. Implement an application that simulates a basic computer that can perform +,-,/,* operations;
    a. User will insert from keyboard the operation that he wants to perform
    b. User insert first number;

```java
public class M_Scanner {
    public static void main(String args[]){
        // Declare the object and initialize with
        // predefined standard input object
        Scanner sc = new Scanner(System.in);

        // String input
        System.out.println("Insert a name");
        String name = sc.nextLine();

        // Character input
        System.out.println("Insert gender M/F");
        char gender = sc.next().charAt(0);

        // Numerical data input
        // byte, short and float can be read
        // using similar-named functions.
        System.out.println("Insert the age");
        int age = sc.nextInt();

        System.out.println("Insert the mobile number");
        long mobileNo = sc.nextLong();

        // Print the values to check if input was correctly obtained.
        System.out.println("Name: "+ name);
        System.out.println("Gender: "+ gender);
        System.out.println("Age: " + age);
        System.out.println("Mobile Number: "+ mobileNo);
    }
}
```

c. User insert second number;

d. System display the calculation results;

2. Implement an application that calculates the sum of two very big numbers

   a. Read the numbers using Scanner class

   b. Hints:

      i. Use StringBuilder, chartAt;

      ii. Check what number is the longest and add padding for the other one?

      iii. Use Integer instead of int, Integer.ParseInt;

      iv. Use String.valueOf(), String.length();

First number: 123612873618273612863781212323234

Second number: 123612873618273612863781212323234

Max Length: 33

The sum!!!!!

Final Result: 1236128748544023490465173409610 46