Java: Beginning

Computer Engineering Department

Java Programming Course

Prof. Dr. Ahmet Sayar

Kocaeli University - Fall 2021

JAVA

- Platform independent
- Compile once run everywhere
- JVM - Java Virtual Machine

Applications and Applets

- Two kinds of Java programs: *applications* and *applets*
- Applications
  - regular programs
  - meant to be run on your computer
- Applets
  - little applications
  - meant to be sent to another location on the Internet and run there

Class Structure

```
package myprojects.javaprojects.prj1;
import java.util.*;

public class Test {
        int x;
        int y;

        public static void main(String args[]){
                int a;
                int b;
        }
}
```

Program Structure

```
class CLASSNAME {
    public static void main(String[] arguments) {
        STATEMENTS
    }
}
```

First Program

```
class Hello {
    public static void main(String[] arguments) {
        // Program execution begins here
        System.out.println("Hello world.");
    }
}
```

Some Terminology

- The person who writes a program is called the *programmer.*
- The person who interacts with the program is called the *user.*
- A *package* is a library of classes that have been defined already.
  - `import java.util.*`

Some Terminology, cont.

- The item(s) inside parentheses are called *argument(s)* and provide the information needed by methods.
- A *variable* is something that can store data.
- an instruction to the computer is called a *statement*; it ends with a semicolon.
- The grammar rules for a programming language are called the *syntax* of the language.

Printing to the Screen

```
System.out.println ("Whatever you want to print");
```

- `System.out` is an object for sending output to the screen.
- `println` is a method to print whatever is in parentheses to the screen.
- How about System.out.print

- int var=3;
- System.out.println(var + " time A"); ?

Printing to the Screen

```
class Hello2 {
    public static void main(String[] arguments) {
        System.out.println("Hello world.");  // Print once
        System.out.println("Line number 2");  // Again!
    }
}
```

Simple Input

- Sometimes the data needed for a computation are obtained from the user at run time.
- Keyboard input requires

  ```
  import java.util.*;
  ```

  at the beginning of the file.

java.util

- import java.util.Arrays;
- import java.util.ArrayList;
- import java.util.Calendar;
- import java.util.Currency;
- import java.util.Date;
- import java.util.HashMap;
- import java.util.LinkedList;
- import java.util.Random;
- import java.util.Scanner;
- import java.util.StringTokenizer;
- import java.util.Vector;

Simple Input, cont.

- Data can be entered from the keyboard using

```
Scanner keyboard =
        new Scanner(System.in);
```

followed, for example, by

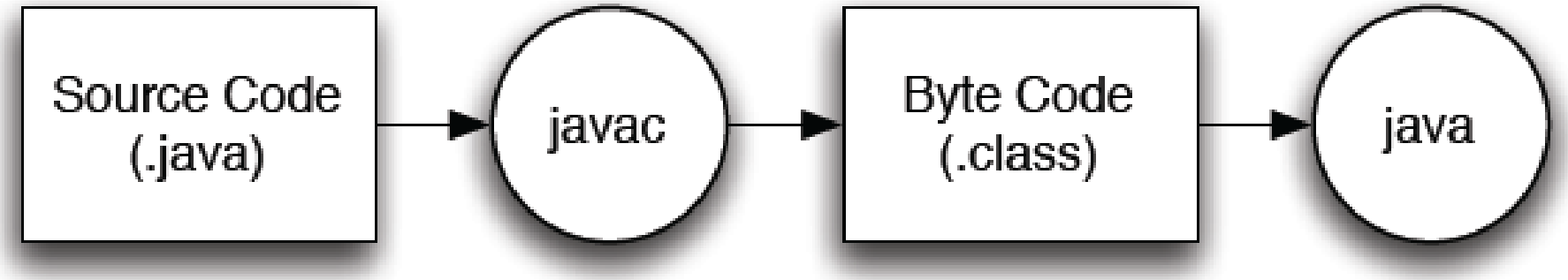
```
int eggsPerBasket = keyboard.nextInt();
double d1 = keyboard.nextDouble();
```

which reads one `int` value from the keyboard and assigns it to `eggsPerBasket`.

Compiling a Java Program or Class

- A Java program consists of one or more classes, which must be compiled before running the program.
- You need not compile classes that accompany Java (e.g. `System` and `Scanner`).
- Each class should be in a separate file.
- The name of the file should be the same as the name of the class.

Compiling



Compiling and Running

- Use an *IDE* (integrated development environment) which combines a text editor with commands for compiling and running Java programs.
- When a Java program is compiled, the byte-code version of the program has the same name, but the ending is changed from `.java` to `.class`.

Compiling and Running, cont.

- A Java program can involve any number of classes.
- The class to run will contain the words

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

near the beginning of the file.

Java Version History

- JDK Alpha and Beta (1995)
- JDK 1.0 (January 23, 1996)
- JDK 1.1 (February 19, 1997)
- J2SE 1.2 (December 8, 1998)
- J2SE 1.3 (May 8, 2000)
- J2SE 1.4 (February 6, 2002)
- J2SE 5.0 (September 30, 2004)
- Java SE 6 (December 11, 2006)
- Java SE 7 (July 28, 2011)
- **Java SE 8 (March 18, 2014) (Extended Suppor until March 2025)**
- Java SE 9 (September 21, 2017) (No extended support)
- Java SE 10 (March, 20, 2018) (No extended support)
- Java SE 11 – September 2018 (open to free public updates)
- Java SE 12 – March 2019 (open to free public updates)
- Java SE 13 – September 2019 (open to free public updates)

Netbeans

- Roman Staněk formed a company around the project and produced commercial versions of the NetBeans IDE
- Netbeans was bought by Sun Microsystems in 1999. Sun open-sourced the NetBeans IDE in June of the following year.
- In 2010, Sun (and thus NetBeans) was acquired by Oracle Corporation.
- In September 2016, Oracle submitted a proposal to donate the NetBeans project to the Apache Software Foundation,

Netbeans

- **NetBeans IDE 8.2** was released on 3 October 2016.
- Netbeans 9.0, which adds support for Java 9 and 10, was released on 29 July 2018, by the Apache Incubator project.
- NetBeans 10.0 was released on 27 December 2018. It brings support for Java 11 and improved support for PHP (7.0–7.3).
- NetBeans 11.1 was released on 22 July 2019.

What does Deprecate Mean in Java?

- When you design an API, carefully consider whether it supersedes an old API. If it does, and you wish to encourage developers (users of the API) to migrate to the new API, then deprecate the old API. Valid reasons to deprecate an API include:
  - It is insecure, buggy, or highly inefficient
  - It is going away in a future release
  - It encourages bad coding practices
- Deprecation is a reasonable choice in all these cases because it preserves "backward compatibility" while encouraging developers to change to the new API. Also, the deprecation comments help developers decide when to move to the new API, and so should briefly mention the technical reasons for deprecation.

Input from Keyboard - 1

```
import java.util.*;

public class FirstProgram
{
    public static void main(String[] args)
    {
        System.out.println("Hello out there.");
        System.out.println("I will add two numbers for you.");
        System.out.println("Enter two whole numbers on a line:");

        int n1, n2;

        Scanner keyboard = new Scanner(System.in);
        n1 = keyboard.nextInt();
        n2 = keyboard.nextInt();

        System.out.println("The sum of those two numbers is");
        System.out.println(n1 + n2);
    }
}
```

Name of the package (library) that includes the `Scanner` *class.*

Name of the program.

Sends output to screen.

Says that `n1` *and* `n2` *are variables that hold integers (whole numbers)*

Sets up things so the program can have keyboard input.

Reads one whole number from the keyboard

Sample Screen Dialog

```
Hello out there.
I will add two numbers for you.
Enter two whole numbers on a line:
12 30
The sum of those two numbers is
42
```

Display 1.5

A Sample Java Program

Input from Keyboard - 2

```
import java.util.*;

public class EggBasket2
{
    public static void main(String[] args)
    {
        int numberOfBaskets, eggsPerBasket, totalEggs;

        Scanner keyboard = new Scanner(System.in);
        System.out.println("Enter the number of eggs in each basket:");
        eggsPerBasket = keyboard.nextInt();
        System.out.println("Enter the number of baskets:");
        numberOfBaskets = keyboard.nextInt();

        totalEggs = numberOfBaskets * eggsPerBasket;

        System.out.println("If you have");
        System.out.println(eggsPerBasket + " eggs per basket and");
        System.out.println(numberOfBaskets + " baskets, then");
        System.out.println("the total number of eggs is " + totalEggs);

        System.out.println("Now we take two eggs out of each basket.");

        eggsPerBasket = eggsPerBasket - 2;
        totalEggs = numberOfBaskets * eggsPerBasket;

        System.out.println("You now have");
        System.out.println(eggsPerBasket + " eggs per basket and");
        System.out.println(numberOfBaskets + " baskets.");
        System.out.println("The new total number of eggs is "
                                                + totalEggs);
    }
}
```

Name of the package (library) that includes the Scanner *class.*

Sets up things so the program can have keyboard input.

Reads one whole number from the keyboard

Sample Screen Dialog

```
Enter the number of eggs in each basket:
6
Enter the number of baskets:
10
If you have
6 eggs per basket and
10 baskets, then
the total number of eggs is 60
Now we take two eggs out of each basket.
You now have
4 eggs per basket and
10 baskets.
The new total number of eggs is 40
```

Display 2.3

Program with Keyboard Input

Input from Command Line

```
package Lab_1;

public class HelloWorld {

	public static void main (String args[]) {
		System.out.println("Hello World! ");
		System.out.println(args[0]);
	}
}
```

Since it is in package Lab_1, it is expected to be located in [project_path]/src/Lab_1/HelloWorld.java

Sample Command Lines Compiling and Running

- [project_path]>javac Lab_1/HelloWorld.java
  - HelloWorld.class is created
  - This is bytecode class or also called target class
- [project_path]>java Lab_1/HelloWorld Ahmet
  - Output: ???
  - Hello World!
  - Ahmet
- [project_path]>java Lab_1/HelloWorld Ahmet Sayar
  - Output: ???
  - How about output of System.out.println(args[2]); ???

Questions?

- What is JVM?
- What is byte code, source code, target code
- Is java portable how?
- What is java SDK?
- What is java JRE?
- Not specifically for java, in general
  - What is linking?
  - What is loading?
- What is compiling (derleyici)?
- What is interpreting (yorumlayici)?
- Is JAVA compiler based or interpreter based programming language? How?

Questions?

- What is the extension of sourcecode java class
- What is the extension of bytecode (targetcode) java class
- What is the command to compile a java class?
- What is the command to run a java class?
- Can java run on every platform?
- What is the name of small java codes embedded in html and run on web environment?
- What is encapsulation?