1DV532 – Starting Out with Java

Lesson 8

Java Class Library and Packages

Dr. Nadeem Abbas

nadeem.abbas@lnu.se

Linnæus University

Java Class Library

- Java comes with a gigantic class library that provides more than 6000 pre-written classes.

  - The idea of class library is to provide a pre-developed, well tested, and easy to use classes that model frequently used objects and operations.
    - For example, classes to perform common mathematical operations, data search and sort operations, etc.

  - Use of the library classes enables programmers to speed up and improve quality of the programming process.
    - Instead of reinventing a wheel, one can simply reuse classes with high quality of code from the Java Class Library

Packages

- Classes in the Java Class Library are organized into *packages*
- A **package** is a collection of related classes that are grouped together. It works like a folder in a file directory.
- Package name is an essential part of the fully-qualified name of a class
- **Fully-Qualified Class Name** is a class name preceded by the package name that contains the class.
  - General form of the fully-qualified class name is as follows:
    - packageName.className,
      - For example, *java.util.*Scanner, here Scanner is the class name and *java.util* is the package name that contains the java library class String.
- Java packages are divided into two categories:
  1. **Built-in Packages:**
  2. **User-defined Packages**

Built-in Packages

- These are the packages provided by the Java Class Library.
- A few of the commonly used packages and their classes are as follows:
  - ***`java.lang`***
    - contains classes such as *String, System, Object*, etc.
  - ***`java.util`***
    - contains utility classes such as *ArrayList, Date, Scanner*, etc.
  - ***`java.io`***
    - contains classes related to system input and output such as *Reader, Writer, File*, etc.
  - ***java.net***
    - Provides the classes for implementing networking applications.

Creating User-defined Packages

- As the name indicates, these are the packages created by programmers using *package* statement as follows:
  - `package packagename;`
- **To create a package:**
  - choose a meaningful name for the package
    - Package names are written in all lower case letters, e.g. mypackage, wo222ab_assign1, dv532.st19.step2, etc.
  - put a package statement with package name at the top of every source (.java) file that you want to include in the package.
    - The package statement, for example, `package wo222ab_assign1;` must be the first line in the source file.
    - There can be only one package statement in one source file.
- Person class that we use as an example in this lesson contains a packages statement: **`package dv532.st19.step2;`**
  - Using this package statement, we make Person class to be part of a package named *dv532.st19.step2*

Using Package Members

- Classes that comprise a package are known as the **package members**.

- To use a package member such as Class from outside its package, i.e., in another package, we need to do one of the following:

  1. **Refer to the Class by its fully qualified name**
     - For example, `java.util.Scanner;`
  2. **Import the package member**
     - `import java.util.Scanner; // import statement`
  3. **Import the member's entire package**
     - `import java.util.*; // import statement.`

- *java.lang* is a default package. All its members are automatically imported.
- To use classes from all other packages, we have to follow one of the above specified methods.

Application Programming Interface – API

- Java Class Library is commonly referred as Java **Application Programming Interface (API)**
- Java provides an API documentation which can be accessed online at https://docs.oracle.com/en/java/javase/12/docs/api/index.html
- The API documentation is a great resource to explore, learn, and use the Java library classes.
- A few of the useful classes to look and play with are
    - java.lang.String
    - java.util.Arrays
    - java.util.ArrayList
    - java.util.Random
    - java.util.Scanner
    - java.lang.Math
    - java.util.Date
    - java.io.File

String Class

- **String** is a java library class placed in **java.lang** packagae.
- It is used to represent Strings that are sequence of characters written within double quotes, e.g., “Hello” is a string of 5 characters.

- The String objects can be created in two ways:
  1. String city = “Lund”;
  2. String city = new String (“Lund”);
  - A difference between the two is that using the first way, if there already exists a String Object with same value, Java compiler does not create a new Object, rather it assigns reference of the already existing object to the new instance variable. But using the second way, a new object is always created even if there already exists a String object with same value.

String Class Methods - A few of the useful String methods are listed below, see the Java API documentation for complete details.

Return Type	Method	Description
char	charAt(int index)	Returns the char value at the specified index.
String	concat(String str)	Concatenates the specified string to the end of this string.
int	length()	Returns the length of this string.
boolean	contains (CharSequence s)	Returns true if and only if this string contains the specified sequence of char values.
boolean	equals (Object anObject)	Compares this string to the specified object.
boolean	startsWith(String prefix)	Tests if this string starts with the specified prefix.
boolean	endsWith (String suffix)	Tests if this string ends with the specified suffix.
String	substring (int beginIndex)	Returns a string that is a substring of this string beginning with a character at the specified index.
String	substring (int beginIndex, int endIndex)	Returns a string that is a substring of this string beginning at the specified beginIndex till endIndex - 1.
String	toLowerCase()	Converts all of the characters in this String to lower case.
String	toUpperCase()	Converts all of the characters in this String to upper case.

Example program, ***StringDemo.java***, demonstrates use of the String class methods.

Math Class

- **Math** is a java library class placed in **java.lang** packagae.
- It provides useful methods for performing basic numeric operations such as finding maximum, minimum, square, square root, logarithm, and trigonometric functions.
- A few of the useful Math class methods are listed on next slide, see the Java API documentation for complete details.
- Example program, ***MathDemo.java***, demonstrates use of the String class methods.

Math Class Methods

Return Type	Method	Description
int	abs (int a)	Returns the absolute value of an int value.
double	ceil (double a)	Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer.
double	floor (double a)	Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.
int	max (int a, int b)	Returns the greater of two int values.
int	min (int a, int b)	Returns the smaller of two int values.
double	pow (double a, double b)	Returns the value of the first argument raised to the power of the second argument.
double	sqrt (double a)	Returns the correctly rounded positive square root of a double value
double	log10 (double a)	Returns the base 10 logarithm of a double value.
double	random()	Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.

Random Class

- **Random** is a java library class placed in package **java.util**.
  - To use this class, you have to explicitly import it as follows:
    - import java.util.Random;

- It provides useful methods to generate *int, long, double, float, and boolean* type random value.
  - A few of these methods are listed on next slide, see the Java API documentation for complete details

- Example program, ***RandomDemo.java***, demonstrates use of the Random class methods.

Random Class Methods

Return Type	Method	Description
boolean	nextBoolean()	Returns a random boolean value
void	nextBytes (byte[] bytes)	Generates random bytes and places them into a user-supplied byte array.
int	nextInt()	Returns a random int value
int	nextInt (int n)	Returns a random int value between 0 and n.
long	nextLong()	Returns a random long value
float	nextFloat()	Returns a random float value between 0.0F and 1.0F
double	nextDouble()	Returns a random double type value between 0.0 and 1.0

Data Class

- **Date** is a java library class placed in package **java.util**.
  - To use this class, you have to explicitly import it as follows:
    - import java.util.Date;
- It is used to represent a specific instant in **time**, with millisecond precision.
- A Date object is created using a new operator as follows:

```
Date today = new Date (); // creates a date object
                //initialized with current date and time.
```

- A few of the Date class methods are listed below, see the Java API documentation for complete details.
  - `long getTime( )`
  - `void setTime(long time)`
  - `boolean after(Date date)`
  - `boolean before(Date date)`
  - `int compareTo(Date date)`
- Example program, ***DateDemo.java***, demonstrates use of the Date class methods.

Suggested Readings

- Absolute Java, Global Edition, 6/E by Walter J. Savitch, Chap 5 Defining Classes II, Section 5.4 “Packages and Javadoc”
- Introduction to Java Programming, Brief Version, Global Edition, 11/E Liang, Chapter 9 “Objects and Classes”, Section 9.6, Using Classes from the Java Library

- Java Tutorial – Packages at https://docs.oracle.com/javase/tutorial/java/package/index.html

- Java API Documentation at https://docs.oracle.com/en/java/javase/14/docs/api/index.html

Lnu.se