Programming 1

Lecture 3 – Data types & Variables 2
Standard Input & Basic Loop

Faculty of Information Technology
Hanoi University

Contents

- **char** and **String**
- type casting
- Java naming convention
- Standard input
- For loop

`char` data type

- **Memory size:** 2 bytes (0 to 65,535)
- Can store *char literals* as well as *integers*.

```
char c = 'Q'; // ASCII
```

```
char x = 'Ă'; // unicode
```

```
char x = 65; // A
```

```
char x = 1260; // Ӭ
```

char and String

- A **String** is composed of **char**acters

```
String s = "HELLO";
```

String literal


```
int len = s.length();
```

(the length is **5**)

```
char c = s.charAt(0);
```

```
char c2 = s.charAt(1);
```

String type is not primitive

- String variables and literals are **object**s
- An object has members (*attributes*, *operators*) which can be accessed through a "dot".
- Examples
  - `System.out`: `System` is an object, `out` is one of its attributes, and also an object itself.
  - `Math.sqrt()`: `sqrt` is an operator which belongs to the `Math` object. Operators carry a function, which is square root calculation in this case.
  - `"abc".charAt(1)`: get character `'b'` from the String.

String operations

- Java supports many operations on a String

```
String str = "Programming";
```

```
String s = str.toLowerCase();
```

```
"programming"
```

```
String s = str.toUpperCase();
```

```
"PROGRAMMING"
```

```
String s = str.replace("m","n");
```

```
"Progranning"
```

```
String s = str.substring(1, 4);
```

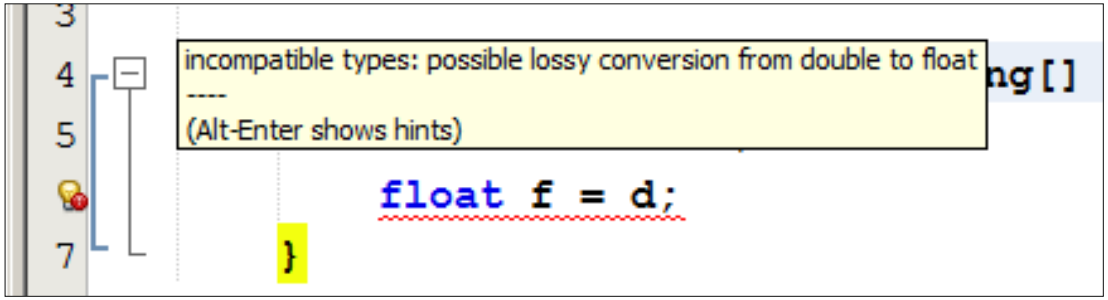
```
"rog"
```

See https://docs.oracle.com/javase/7/docs/api/java/lang/String.html

Type casting (conversion)

- Converting a value from one data type to another.

```
double d = 1.5;
float f = d;
```




```
double d = 1.5;
float f = (float) d;
```

Type casting

- Automatic (implicit) type casting

```
int a = 15;
long b = a;
```

Byte –> Short –> Int –> Long – > Float –> Double

Widening or Automatic Conversion

```
float f = 2.1f;
double d = f; // no error
```

Type casting

- Explicit type casting

```
long a = 15;
int b = (int) a;
```

Double –> Float –> Long –> Int –> Short –> Byte

Narrowing or Explicit Conversion

```
double d = 2.1;
float f = (float) d;
```

Type casting

- Get number values from String

```
String s = "15"; // number stored as String
int n = Integer.parseInt(s); // convert to int
```

```
String s = "4.8"; // real number as String
float score = Float.parseFloat(s);
```

```
String s = "3.5";
double score = Double.parseDouble(s);
```

```
String s = "467837482343";
long n = Long.parseLong(s);
```

Java naming convention

- To make our codes easier to read

Incosistent naming style

```
public class Fruitprice {
    public static void main(String[] args) {
        int numof_Apples = 3;
        int NumOforanges = GET_Intfrom_keyboard();
        double toTaLpRiCe = numof_Apples * 30 + NumOforanges * 15;
        System.out.println(toTaLpRiCe);
    }
}
```

Java naming convention

- …and more beautiful

Consistent naming style

```
public class FruitPrice {
    public static void main(String[] args) {
        int numOfApples = 3;
        int numOfOranges = getIntFromKeyboard();
        double totalPrice = numOfApples * 30 + numOfOranges * 15;
        System.out.println(totalPrice);
    }
}
```

Java naming convention

- Class names: UpperCamelCase
  - E.g. `Sprite`, `ImageSprite`, `StudentManager`
- Methods: lowerCamelCase
  - E.g. `age()`, `myAge()`, `getMyAge()`
- Variables: lowerCamelCase
  - E.g. `height`, `maxHeight`, `errorMessage`
- `Constants: uppercase & underscore`
  - E.g. `ROWS_PER_PAGE`, `PLATFORM`, `WINDOWS_WIDTH`

Example 1

- Write the code to extract any of the following class names

  `1C17, 3K16, 8A18`

  into 3 parts:
  - the class number (1, 3, 8), an integer
  - the department code (C, K, A), a String
  - the year (17, 16, 18), an integer

Answer

```
public class Example1 {
	public static void main(String[] args) {
		String s = "1C17";
		String s1 = s.substring(0, 1);
		int classNumber = Integer.parseInt(s1);
		String className = s.substring(1, 2);
		String s2 = s.substring(2, 4);
		int classYear = Integer.parseInt(s2);
	}
}
```

String comparison

```
1   public class StringCompare {
2       public static void main(String[] args) {
3           String s1 = "B".replace("B", "A");
4           String s2 = "A";
5           if (s1 == s2) {
6               System.out.println("Equal");
7           } else {
8               System.out.println("Not equal");
9           }
10      }
11  }
```

```
Output - PR1 (run)
run:
Not equal
BUILD SUCCESSFUL (total time: 0 seconds)
```

- The == operator is not used for String or object comparison

String comparison

```
1   public class StringCompare {
2       public static void main(String[] args) {
3           String s1 = "B".replace("B", "A");
4           String s2 = "A";
5           if (s1.equals(s2)) {
6               System.out.println("Equal");
7           } else {
8               System.out.println("Not equal");
9           }
10      }
11  }
```

```
Output - PR1 (run)
run:
Equal
BUILD SUCCESSFUL (total time: 0 seconds)
```

- The **equals** operator/method should be used instead.

`if…else…if` statement

```
// find largest among a, b and c
if (a > b && a > c) {
    System.out.println(a + "  is largest");
} else if (b > c) {
    System.out.println(b + " is largest");
} else {
    System.out.println(c + " is largest");
}
```

- The conditional statement **`if`** can be chained to divide the logic into many branches.

Example 2

- Given a student's score **S** $\in [0, 10]$. Show his grade letter based on these rules:
  - If $0 \leq S < 6$, grade **F**
  - If $6 \leq S < 7$, grade **D**
  - If $7 \leq S < 8$, grade **C**
  - If $8 \leq S < 9$, grade **B**
  - If $9 \leq S \leq 10$, grade **A**
- For $S = 7$, expected result:

```
Score: 7.0. Grade: C
```

Answer

```
double s = 7; // assuming s is in valid range
System.out.print("Score: " + s + ". ");
if (s < 6)
   System.out.println("Grade: F");
else if (s < 7) // means: s >= 6 && s < 7
   System.out.println("Grade: D");
else if (s < 8)
   System.out.println("Grade: C");
else if (s < 9)
   System.out.println("Grade: B");
else
   System.out.println("Grade: A");
```

The `import` statement

- Import class(es) into a program so that we can use them in our code.

```
import java.util.Scanner;
public class MyApp {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
   }
}
```

The `Scanner` class

- `System.in` represents the keyboard.
  (And `System.out` represents the monitor/screen)

```
import java.util.Scanner;
public class MyApp {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
   }
}
```

Get an integer with Scanner

- Get an integer from keyboard with Scanner.
- Use Scanner's `nextInt()` method or `nextLong()` method.

```
import java.util.Scanner;
public class MyApp {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int a = sc.nextInt();
      long b = sc.nextLong();
   }
}
```

Get a real number with Scanner

- Use Scanner's `nextDouble()` method or `nextFloat()` method.

```
import java.util.Scanner;
public class MyApp {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      double d = sc.nextDouble();
      float f = sc.nextFloat();
   }
}
```

Get a String with Scanner

- Use Scanner's `nextLine()` method to get a line of text.
- Use Scanner's `next()` method to get one word.

```
import java.util.Scanner;
public class MyApp {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String s = sc.nextLine();
      String w = sc.next();
   }
}
```

Abstract vs. Primitive data types

- **Primitive:** values directly map to machine representations.

```
int a = 5;
```

- **Abstract:** actual representation is hidden. Only a public API is exposed.

```
Scanner sc = new Scanner(System.in);
int n = sc.nextInt(); // public method
```

- Abstract data types are represented by classes in Java.

Don't forget to interact with the user

- It is customary to print a text before you let user enter something.

```
import java.util.Scanner;
public class MyApp {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter ur name: ");
      String s = sc.nextLine();
   }
}
```

Get multiple values at the same time

- In this case, the user just has to enter 3 numbers in one line, separated by spaces.

```
import java.util.Scanner;
public class MyApp {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter 3 numbers: ");
      int a = sc.nextInt();
      double b = sc.nextDouble();
      int c = sc.nextInt();
   }
}
```

Validation

- We always want to make sure a variable has a desired value.

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter year of birth: ");
int yob = sc.nextInt();
int age = 2018 - yob;
System.out.print("Your age is: " + age);
```

Result

```
Enter year of birth: 2048
Your age is: -30
```

Validation

- We never know what a user types in.

Console

```
Welcome to the bank, user!
Your balance is: 300
How much do you want to withdraw? -5000
Thank you!
Your balance is now: 5300
```

Cases of validation

- Check if X equals some value, for example:
  - Get the option that a user chooses from a menu
  - Checking a password
  - Check if a String has a certain length (often in encryption problem)
- Check if X is one of several values
  - A grade letter should exists in the set {A, B, C, D, F}
  - A math operator should be one of the following: + - * /

Cases of validation

- Check if X is between some range
  - Example: Age or Cost shouldn't be negative.
- Check if X has certain characteristics
  - A name string shouldn't be empty.
  - A string shouldn't be too long. Otherwise it would be inappropriate or there wouldn't be enough memory to save it. Better check its length.
  - Check for a valid email address or phone number.

Example 3

- Write a program to ask user to enter a password.
- If he enters the correct password (which is “abc123"), then print a "Login successful" message.

Expected output

```
Enter the password: abc123
Login successful!
```

Answer

```
import java.util.Scanner;
public class Login {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the password: ");
        String password = sc.nextLine();
        if (password.equals("abc123")) {
            System.out.println("Login successful!");
        }
    }
}
```

Example 4

- Ask the user for his age.
- If his age is less than 13, print message "Not for kids", or if it's greater than 19, print "You're too old"

Expected output

```
How old are you? 20
You're too old
```

Expected output

```
How old are you? 12
Not for kids
```

Answer

```
import java.util.Scanner;
public class AgeRestriction {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("How old are you? ");
        int age = sc.nextInt();
        if (age > 19) {
            System.out.println("You're too old!");
        } else if (age < 13) {
            System.out.println("Not for kids!");
        } else {
            System.out.println("You're welcome!");
        }
    }
}
```