# Chapter 4 – Fundamental Data Types

Dr Kafi Rahman, PhD
Email: kafi@truman.edu
Truman State University

# IntelliJ IDEA

- If students are familiar with JetBrains IDE's like PyCharm, they can consider using IntelliJ IDEA as their code editor of choice.
- Students can get a free license through their Truman student email account for a year

- Visit the following link for more information
  - https://blog.jetbrains.com/education/2018/09/18/free-jetbrains-licenses-as-part-of-github-student-developer-pack/

# How to export a project

- Demo

# Self Check 4.3

- Which of the following initializations are incorrect, and why?
  - `int dollars = 100.0;`
  - `double balance = 100;`
- Answer: The first initialization is incorrect. The right hand side is a value of type double, and it is not legal to initialize an int variable with a double value. The second initialization is correct — an int value can always be converted to a double.

# Arithmetic Operators

- Four basic operators:
- addition: + subtraction: - multiplication: * division: /
- Expression: combination of variables, literals, operators, and/or method calls
  - (a + b) / 2
- Parentheses control the order of the computation
  - (a + b) / 2
- Multiplication and division have a higher precedence than addition and subtraction
  - a + b / 2
- Mixing integers and floating-point values in an arithmetic expression yields a floating- point value
  - 7 + 4.0 is the floating-point value 11.0

# Increment and Decrement

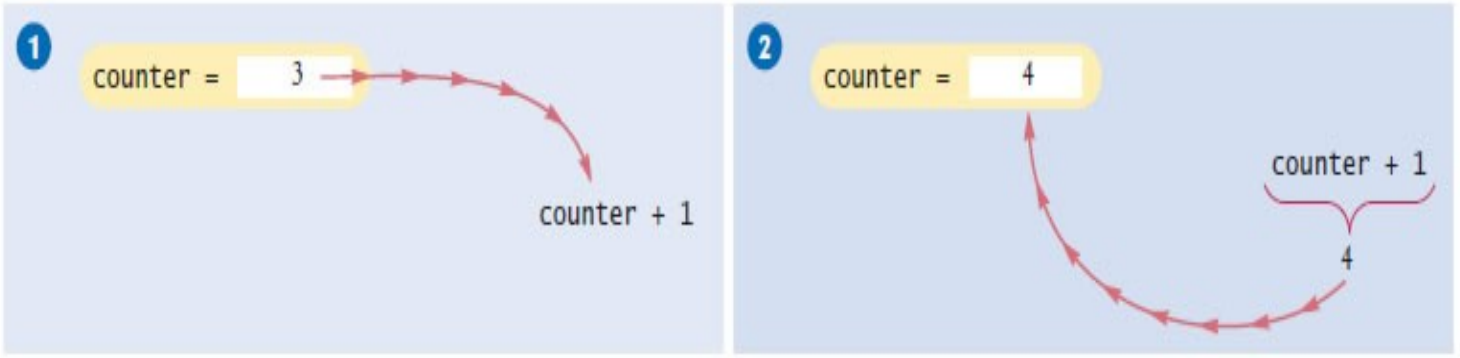



**Figure 1** Incrementing a Variable

- The ++ operator adds 1 to a variable (increments)
- counter++; // Adds 1 to the variable counter
- The -- operator subtracts 1 from the variable (decrements)
- counter--; // Subtracts 1 from counter

# Integer Division and Remainder

- Division works as you would expect, as long as at least one of the numbers is a floating-point number.
- Example: all of the following evaluate to 1.75
  - 7.0 / 4.0
  - 7 / 4.0
  - 7.0 / 4
- If both numbers are integers, the result is an integer. The remainder is discarded
  - 7 / 4 evaluates to 1
- Use % operator to get the remainder with (pronounced "modulus", "modulo", or "mod")
  - 7 % 4 is 3

# Integer Division and Remainder

Table 3 Integer Division and Remainder

| Expression (where n = 1729) | Value | Comment |
|---|---|---|
| `n % 10` | 9 | `n % 10` is always the last digit of `n`. |
| `n / 10` | 172 | This is always `n` without the last digit. |
| `n % 100` | 29 | The last two digits of `n`. |
| `n / 10.0` | 172.9 | Because 10.0 is a floating-point number, the fractional part is not discarded. |
| `-n % 10` | -9 | Because the first argument is negative, the remainder is also negative. |
| `n % 2` | 1 | `n % 2` is 0 if `n` is even, 1 or –1 if `n` is odd. |

# Powers and Roots

- Math class contains methods sqrt and pow to compute square roots and powers To take the square root of a number, use Math.sqrt; for example, Math.sqrt(x)
- To compute xn, you write Math.pow(x, n)
- To compute x2 it is significantly more efficient simply to compute x * x
- In Java.
  - $b \times \left(1 + \frac{r}{100}\right)^n$ can be represented as
  - b * Math.pow(1 + r / 100, n)

# Mathematical Methods

Table 4 Mathematical Methods

| Method | Returns | Method | Returns |
|---|---|---|---|
| `Math.sqrt(x)` | Square root of $x$ ($\geq 0$) | `Math.abs(x)` | Absolute value $\lvert x \rvert$ |
| `Math.pow(x, y)` | $x^y$ ($x > 0$, or $x = 0$ and $y > 0$, or $x < 0$ and $y$ is an integer) | `Math.max(x, y)` | The larger of $x$ and $y$ |
| `Math.sin(x)` | Sine of $x$ ($x$ in radians) | `Math.min(x, y)` | The smaller of $x$ and $y$ |
| `Math.cos(x)` | Cosine of $x$ | `Math.exp(x)` | $e^x$ |
| `Math.tan(x)` | Tangent of $x$ | `Math.log(x)` | Natural log ($\ln(x)$, $x > 0$) |
| `Math.round(x)` | Closest integer to $x$ (as a `long`) | `Math.log10(x)` | Decimal log ($\log_{10}(x)$, $x > 0$) |
| `Math.ceil(x)` | Smallest integer $\geq x$ (as a `double`) | `Math.floor(x)` | Largest integer $\leq x$ (as a `double`) |
| `Math.toRadians(x)` | Convert $x$ degrees to radians (i.e., returns $x \cdot \pi/180$) | `Math.toDegrees(x)` | Convert $x$ radians to degrees (i.e., returns $x \cdot 180/\pi$) |

# Arithmetic Expressions

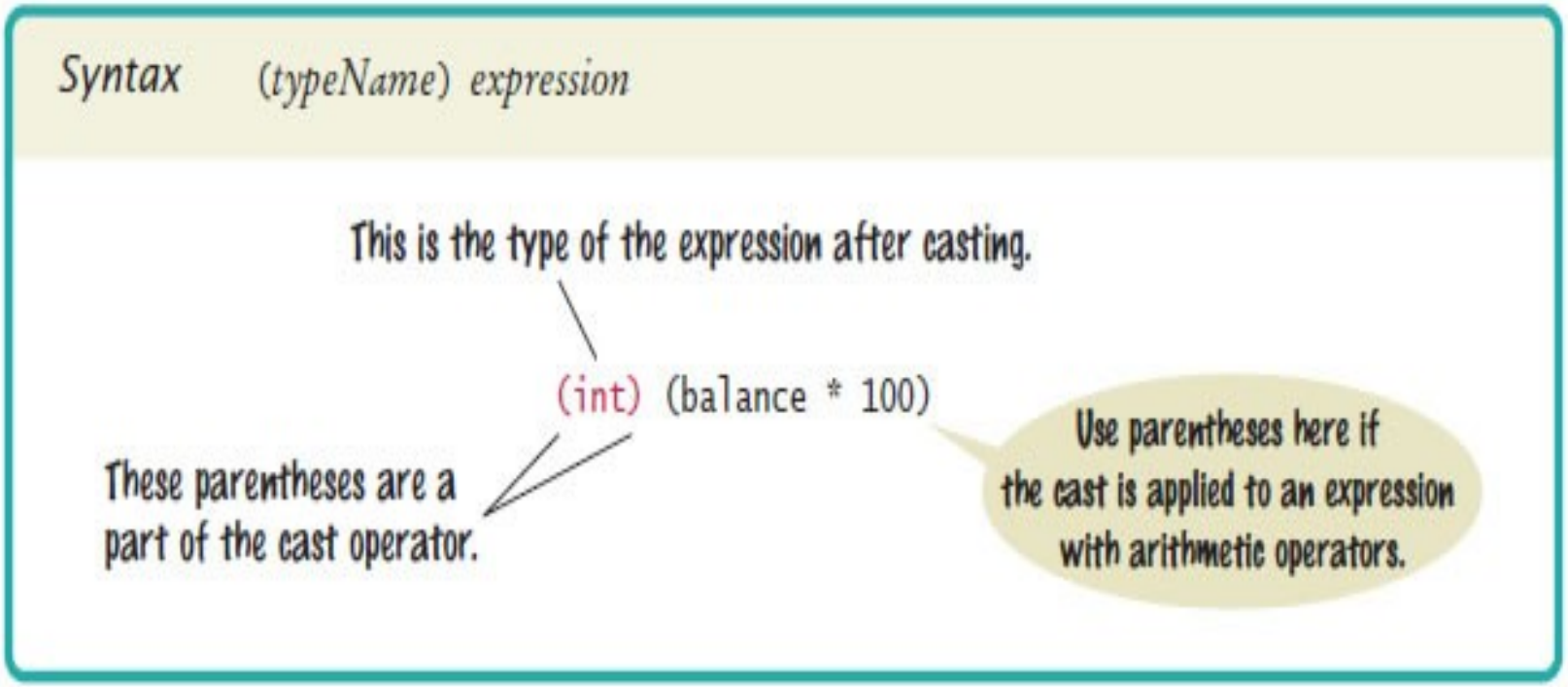
Table 5 Arithmetic Expressions

| Mathematical Expression | Java Expression | Comments |
|---|---|---|
| $\frac{x + y}{2}$ | `(x + y) / 2` | The parentheses are required; `x + y / 2` computes $x + \frac{y}{2}$. |
| $\frac{xy}{2}$ | `x * y / 2` | Parentheses are not required; operators with the same precedence are evaluated left to right. |
| $\left(1 + \frac{r}{100}\right)^n$ | `Math.pow(1 + r / 100, n)` | Use `Math.pow(x, n)` to compute $x^n$. |
| $\sqrt{a^2 + b^2}$ | `Math.sqrt(a * a + b * b)` | `a * a` is simpler than `Math.pow(a, 2)`. |
| $\frac{i + j + k}{3}$ | `(i + j + k) / 3.0` | If $i$, $j$, and $k$ are integers, using a denominator of 3.0 forces floating-point division. |
| $\pi$ | `Math.PI` | `Math.PI` is a constant declared in the `Math` class. |

# Converting Floating-Point Numbers to Integers - Cast

- The compiler disallows the assignment of a double to an int because it is potentially dangerous
  - The fractional part is lost
- The following is an error
  - double balance = total + tax;
  - int dollars = balance; // Error: Cannot assign double to int
- Use the cast operator (int) to convert a convert floating-point value to an integer.
  - double balance = total + tax;
  - int dollars = (int) balance;
- You use a cast (typeName) to convert a value to a different type.

# Syntax 4.2 Cast



# Converting Float to Integers: Rounding Method

- Math.round converts a floating-point number to nearest integer:
  - `long rounded = Math.round(balance);`

- If balance is `13.75`, then rounded is set to `14`.

# Reading Input: Using the Scanner Object

```
import java.util.Scanner; // use object in program

public class CHelloWorld {
     public static void main(String[] args) {

          System.out.println("Welcome to Java!!!");
          // basic input
          Scanner input = new Scanner(System.in);

          System.out.println("Enter the radius of the circle: ");
          double radius = input.nextDouble();

          final double PI = Math.PI;
          double area = PI * Math.pow(radius, 2.0);

          System.out.printf("The area is: %.2f", area); // formatted

          input.close();

     }
}
```

# Formatted Output

- Use the printf method to specify how values should be formatted.
- Printf lets you print this
  - Price per liter: 1.22
- Instead of this
  - Price per liter: 1.215962441314554
- This command displays the price with two digits after the decimal point:
  - System.out.printf("%.2f", price);

# Formatted Output

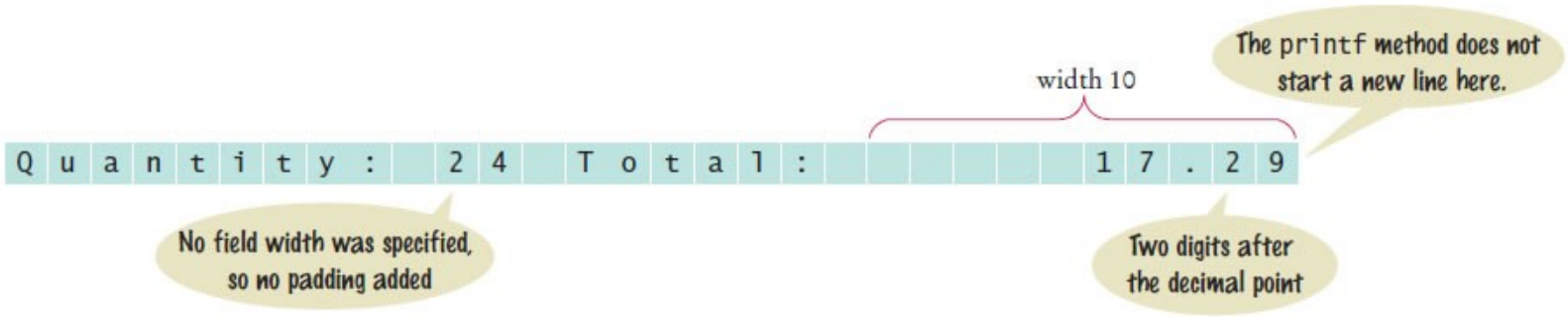

- You can also specify a field width:
  - System.out.printf("%10.2f", price);
  - This prints 10 characters:
    - Six spaces followed by the four characters 1.22

# Formatted Output

Table 6 Format Specifier Examples

| Format String | Sample Output | Comments |
|---|---|---|
| `"%d"` | `24` | Use `d` with an integer. |
| `"%5d"` | `  24` | Spaces are added so that the field width is 5. |
| `"Quantity:%5d"` | `Quantity:  24` | Characters inside a format string but outside a format specifier appear in the output. |
| `"%f"` | `1.21997` | Use `f` with a floating-point number. |
| `"%.2f"` | `1.22` | Prints two digits after the decimal point. |
| `"%7.2f"` | `  1.22` | Spaces are added so that the field width is 7. |
| `"%s"` | `Hello` | Use `s` with a string. |
| `"%d %.2f"` | `24 1.22` | You can format multiple values at once. |

# Formatted Output




- You can print multiple values with a single call to the printf method. Example
  - System.out.printf("Quantity: %d Total: %10.2f", quantity, total);

# String Type

- A string is a sequence of characters.
- You can declare variables that hold strings
  - String name = "Harry";
- A string variable is a variable that can hold a string
- String literals are character sequences enclosed in quotes  A string literal denotes a particular string
- String length is the number of characters in the string
- The length of "Harry" is 5
- The length method yields the number of characters in a string
  - int n = name.length();
- A string of length 0 is called the empty string
  - Contains no characters
  - Is written as ""

# Concatenation

- Concatenating strings means to put them together to form a longer string  Use the + operator:
- Example:
  - String fName = "Harry";
  - String lName = "Morgan";
  - String name = fName + lName;
- Result:
  - "HarryMorgan"

# Concatenation

- If one of the arguments of the + operator is a string
- The other is forced to become to a string: Both strings are then concatenated
- Example
  - String jobTitle = "Agent";
  - int employeeId = 7;
  - String bond = jobTitle + employeeId;
- Result
  - "Agent7"

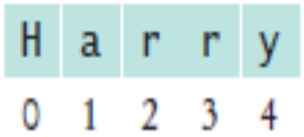
# String Input

- Use the next method of the Scanner class to read a string containing a single word.
- System.out.print("Please enter your name: ");
- String name = in.next();

- Only one word is read.
- Use a second call to in.next to get a second word if needed.

# Strings and Characters

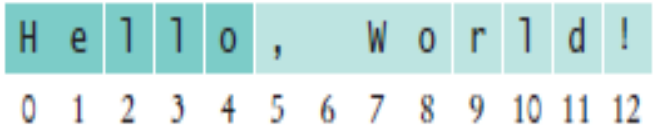
- String positions are counted starting with 0.
- The position number of the last character is always one less than the length of the string.
- The last character of the string "Harry" is at position 4
- The charAt method returns a char value from a string The example
- String name = "Harry";
  - char start = name.charAt(0);
  - char last = name.charAt(4);
  - Sets start to the value 'H' and last to the value 'y'.



# Substrings

- Use the substring method to extract a part of a string. The method call str.substring(start, pastEnd)
- returns a string that is made up of the characters in the string str,
- starting at position start, and
- containing all characters up to, but not including, the position pastEnd.

- Example:
  - String greeting = "Hello, World!";
  - String sub = greeting.substring(0, 5); // sub is "Hello"
- To extract "World"
  - String sub2 = greeting.substring(7, 12);



# Java String Operations

Table 7 String Operations

| Statement | Result | Comment |
|---|---|---|
| `string str = "Ja";`<br>`str = str + "va";` | `str` is set to `"Java"` | When applied to strings, + denotes concatenation. |
| `System.out.println("Please"`<br>`  + " enter your name: ");` | Prints<br>`Please enter your name:` | Use concatenation to break up strings that don't fit into one line. |
| `team = 49 + "ers"` | `team` is set to `"49ers"` | Because `"ers"` is a string, 49 is converted to a string. |
| `String first = in.next();`<br>`String last = in.next();`<br>`(User input: Harry Morgan)` | `first` contains `"Harry"`<br>`last` contains `"Morgan"` | The `next` method places the next word into the string variable. |
| `String greeting = "H & S";`<br>`int n = greeting.length();` | `n` is set to 5 | Each space counts as one character. |
| `String str = "Sally";`<br>`char ch = str.charAt(1);` | `ch` is set to `'a'` | This is a `char` value, not a `String`. Note that the initial position is 0. |
| `String str = "Sally";`<br>`String str2 = str.substring(1, 4);` | `str2` is set to `"all"` | Extracts the substring starting at position 1 and ending before position 4. |
| `String str = "Sally";`<br>`String str2 = str.substring(1);` | `str2` is set to `"ally"` | If you omit the end position, all characters from the position until the end of the string are included. |
| `String str = "Sally";`<br>`String str2 = str.substring(1, 2);` | `str2` is set to `"a"` | Extracts a `String` of length 1; contrast with `str.charAt(1)`. |
| `String last = str.substring(`<br>`  str.length() - 1);` | `last` is set to the string containing the last character in `str` | The last character has position `str.length() - 1`. |

# Self Check 4.22

- What is the length of the string "Java Program"?
- Answer: The length is 12. The space counts as a character.

# Self Check 4.23

- Consider this string variable
- String str = "Java Program";
- Give a call to the substring method that returns the substring "gram".
- Answer: str.substring(8, 12) or str.substring(8)

# Self Check 4.25

- What does the following statement sequence print?

- String str = "Harry";
- int n = str.length();
- String mystery = str.substring(0, 1) + str.substring(n - 1, n);
  System.out.println(mystery);

- Answer: Hy

# Questions?