AP Computer Science A

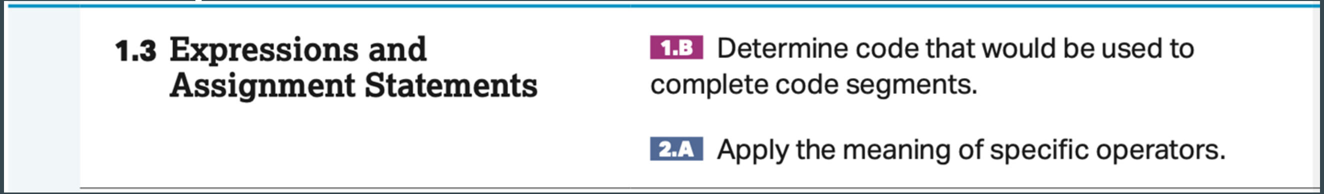
UNIT 1 TOPIC 3
Expressions & Assignment Statements

College Board Alignment

Unit 1 Topic 3

UNIT 1

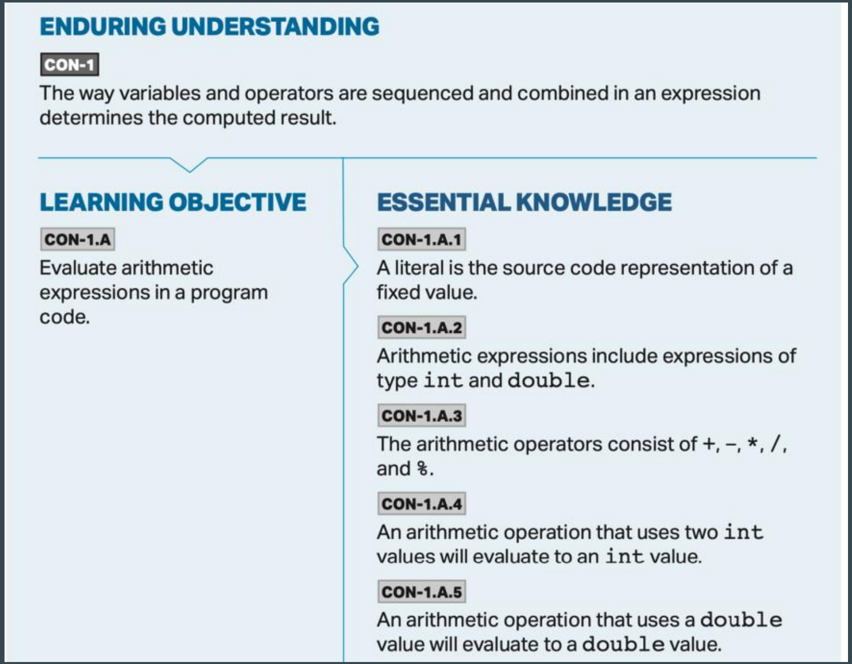
Primitive Types



1.3 Expressions and Assignment Statements

1.B Determine code that would be used to complete code segments.

2.A Apply the meaning of specific operators.



ENDURING UNDERSTANDING

CON-1

The way variables and operators are sequenced and combined in an expression determines the computed result.

LEARNING OBJECTIVE

CON-1.A

Evaluate arithmetic expressions in a program code.

ESSENTIAL KNOWLEDGE

CON-1.A.1

A literal is the source code representation of a fixed value.

CON-1.A.2

Arithmetic expressions include expressions of type `int` and `double`.

CON-1.A.3

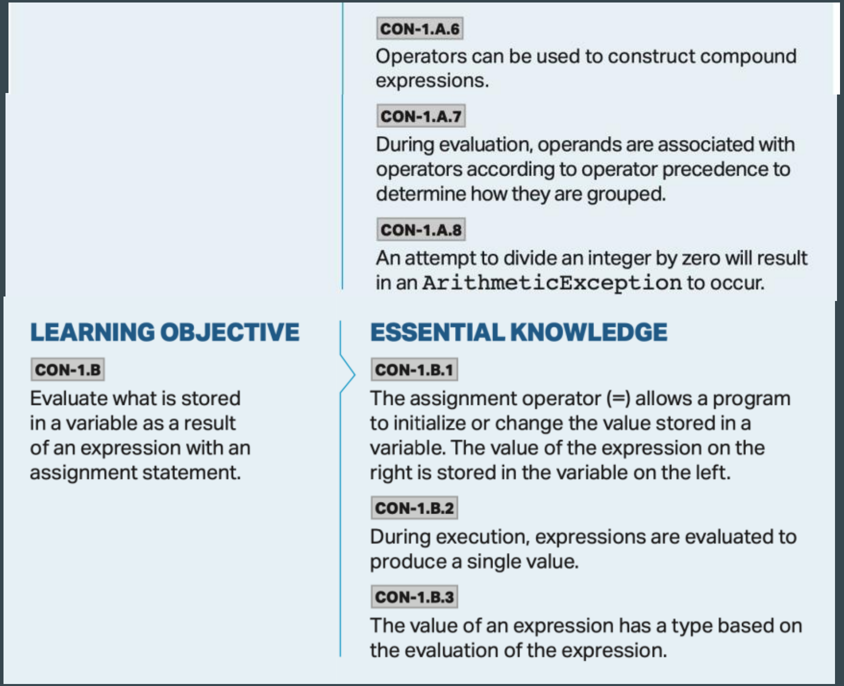
The arithmetic operators consist of +, −, *, /, and `%`.

CON-1.A.4

An arithmetic operation that uses two `int` values will evaluate to an `int` value.

CON-1.A.5

An arithmetic operation that uses a `double` value will evaluate to a `double` value.



CON-1.A.6

Operators can be used to construct compound expressions.

CON-1.A.7

During evaluation, operands are associated with operators according to operator precedence to determine how they are grouped.

CON-1.A.8

An attempt to divide an integer by zero will result in an `ArithmeticException` to occur.

LEARNING OBJECTIVE

CON-1.B

Evaluate what is stored in a variable as a result of an expression with an assignment statement.

ESSENTIAL KNOWLEDGE

CON-1.B.1

The assignment operator (=) allows a program to initialize or change the value stored in a variable. The value of the expression on the right is stored in the variable on the left.

CON-1.B.2

During execution, expressions are evaluated to produce a single value.

CON-1.B.3

The value of an expression has a type based on the evaluation of the expression.

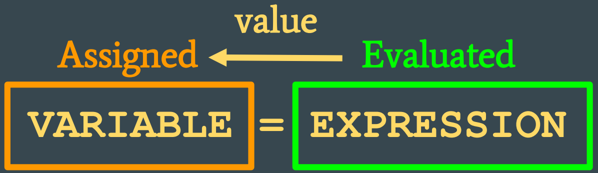
Agenda

- Memory & variables
- Demo: Operators & Mathematical Expressions
- U1T3 Lab (day 1 of 2)

Expressions, Operators & Assignments

Assignment Statements

In an **assignment statement**, the **expression** on the right is *evaluated* and the resulting value is assigned to the variable on the left.



- If the expression on the right is a **literal value**, like `x = 5`, then there is nothing to evaluate and the value of that literal is assigned to the variable on the left.

- If the expression on the right is a **variable**, like `x = y`, then the simple expression `y` is evaluated (which is simply looking up its value) and assigned to the variable on the left (note: "`y`" is not assigned to x, but rather a copy of `y`'s *value*).

- If the expression on the right is a **mathematical expression** (which may involve other variables), like `x = 5 + y` or `x = 2 * (10 - y) + 4`, then the simple expression `y` is evaluated (which is simply looking up its value) and assigned to the variable on the left (note: "`y`" is not assigned to x, but rather `y`'s *value*).

Arithmetic Operators

- The arithmetic operators consist of `+`, `-`, `*`, `/`, and `%` (modulo for the **remainder** in division).

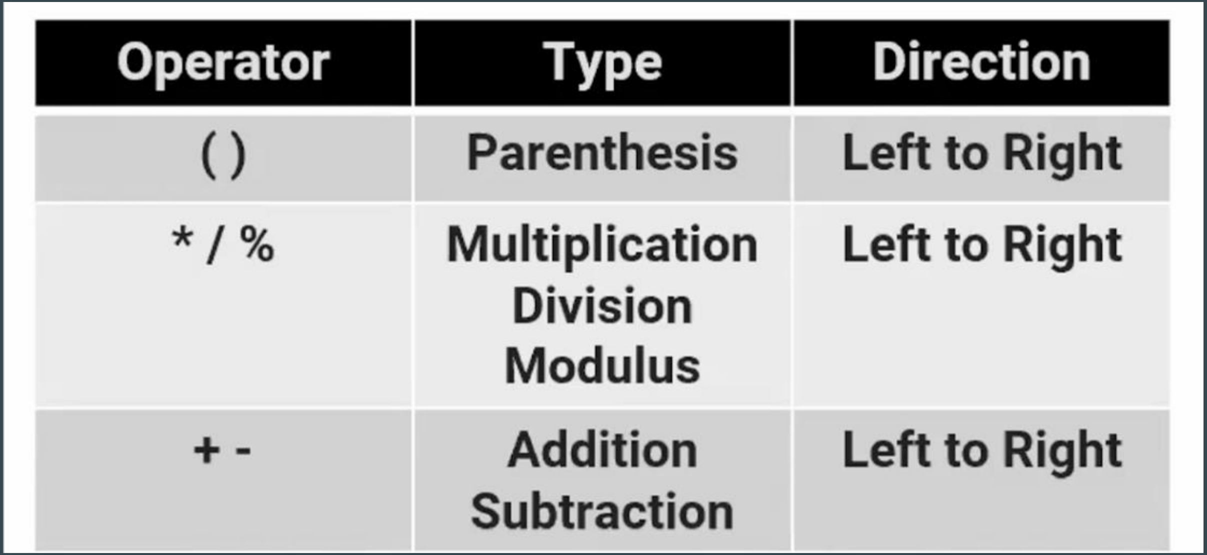
- Recall from last year: `17 % 6 =` **`???`**

Arithmetic Operators

- The arithmetic operators consist of `+`, `-`, `*` , `/` , and `%` (modulo for the **remainder** in division).

- Recall from last year: `17 % 6 = 5  (17 divided by 6 is 2 remainder 5, so 5 is the modulo`

Operator Precedence



Operator	Type	Direction
( )	Parenthesis	Left to Right
* / %	Multiplication Division Modulus	Left to Right
+ -	Addition Subtraction	Left to Right

Example: Arithmetic operators and precedence

- Operators can be used to construct compound expressions.
- During evaluation, operands are associated with operators according to operator precedence to determine how they are grouped. (*, /, % have precedence over + and -, unless parentheses are used to group those.)

```
System.out.println(4 * 2 + 10 / 5 % 2);
```

Example: Arithmetic operators and precedence

- Operators can be used to construct compound expressions.
- During evaluation, operands are associated with operators according to operator precedence to determine how they are grouped. (*, /, % have precedence over + and -, unless parentheses are used to group those.)

```
System.out.println(4 * 2 + 10 / 5 % 2);

8 + 10 / 5 % 2
```

Example: Arithmetic operators and precedence

- Operators can be used to construct compound expressions.
- During evaluation, operands are associated with operators according to operator precedence to determine how they are grouped. (*, /, % have precedence over + and -, unless parentheses are used to group those.)

```
System.out.println(4 * 2 + 10 / 5 % 2);

8 + 10 / 5 % 2

8 + 2 % 2
```

Example: Arithmetic operators and precedence

- Operators can be used to construct compound expressions.
- During evaluation, operands are associated with operators according to operator precedence to determine how they are grouped. (*, /, % have precedence over + and -, unless parentheses are used to group those.)

```
System.out.println(4 * 2 + 10 / 5 % 2);

8 + 10 / 5 % 2

8 + 2 % 2

8 + 0
```

Example: Arithmetic operators and precedence

- Operators can be used to construct compound expressions.
- During evaluation, operands are associated with operators according to operator precedence to determine how they are grouped. (*, /, % have precedence over + and -, unless parentheses are used to group those.)

```
System.out.println(4 * 2 + 10 / 5 % 2);

8 + 10 / 5 % 2

8 + 2 % 2

8 + 0
```

8 gets printed

Example: Arithmetic operators and precedence

Predict the output:

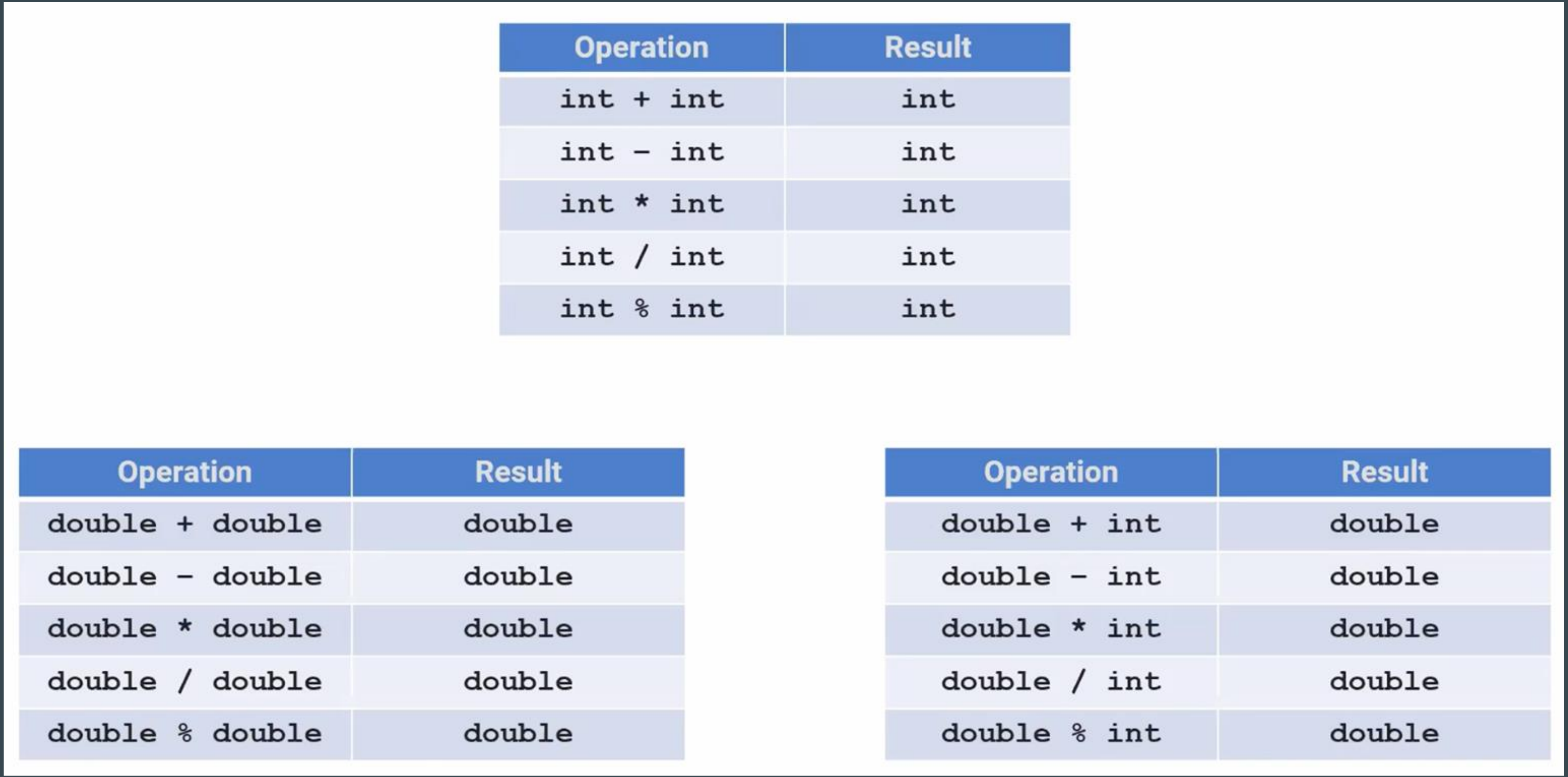
```
System.out.println(3 + 5);
System.out.println(3 + 5.0);
System.out.println(3.0 + 5);
System.out.println(3.0 + 5.0);

System.out.println(3 / 5);
System.out.println(3 / 5.0);
System.out.println(3.0 / 5);
System.out.println(3.0 / 5.0);
```

Dividing By Zero

- An attempt to divide an integer by zero will result in an ArithmeticException to occur, which is a **runtime error** (since the program compiles successfully and is actually off and running), as opposed to a syntax/compiler error (which is a typo that Replit catches and underlines for you)

Summary of operations

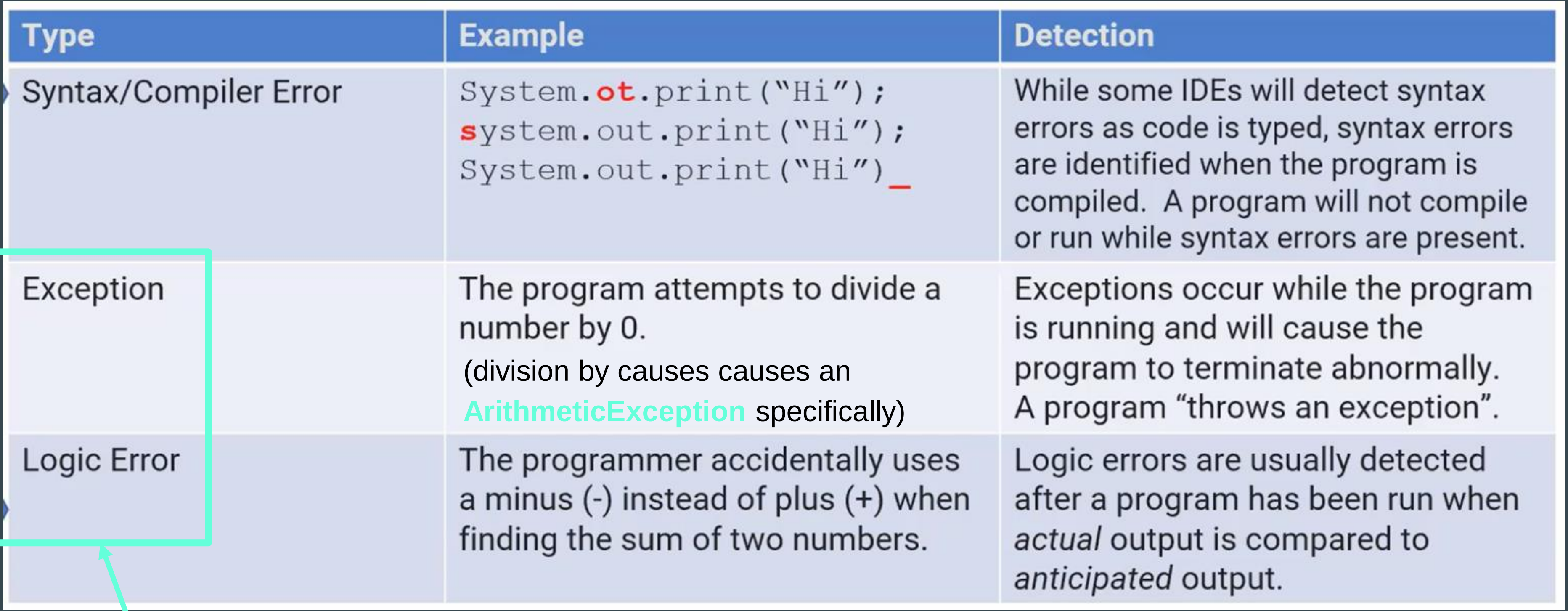


Operation	Result
`int + int`	`int`
`int – int`	`int`
`int * int`	`int`
`int / int`	`int`
`int % int`	`int`

Operation	Result
`double + double`	`double`
`double – double`	`double`
`double * double`	`double`
`double / double`	`double`
`double % double`	`double`

Operation	Result
`double + int`	`double`
`double – int`	`double`
`double * int`	`double`
`double / int`	`double`
`double % int`	`double`

Types of Programming Errors



Type	Example	Detection

Syntax/Compiler Error	`System.ot.print("Hi");`
	`system.out.print("Hi");`
	`System.out.print("Hi")_`
		While some IDEs will detect syntax errors as code is typed, syntax errors are identified when the program is compiled. A program will not compile or run while syntax errors are present.

Exception	The program attempts to divide a number by 0. (division by causes causes an **ArithmeticException** specifically)	Exceptions occur while the program is running and will cause the program to terminate abnormally. A program "throws an exception".
Logic Error	The programmer accidentally uses a minus (-) instead of plus (+) when finding the sum of two numbers.	Logic errors are usually detected after a program has been run when *actual* output is compared to *anticipated* output.

These are both types of "**runtime errors**" since your compiler doesn't catch whatever code causes them (the code is *syntactically* correct)

Summary

- The arithmetic operators consist of +, -, * , / , and % (modulo for the remainder in division).
- An arithmetic operation that uses two int values will evaluate to an int value. With integer division, any decimal part in the result will be thrown away (truncated).
- An arithmetic operation that uses at least one double value will evaluate to a double value.
- Operators can be used to construct compound expressions.
- During evaluation, operands are associated with operators according to operator precedence to determine how they are grouped. (*, /, % have precedence over + and -, unless parentheses are used to group those.)
- An attempt to divide an integer by zero will result in an ArithmeticException to occur.
- The assignment operator (=) allows a program to initialize or change the value stored in a variable. The value of the expression on the right is stored in the variable on the left.
- During execution, expressions are evaluated to produce a single value.
- The value of an expression has a type based on the evaluation of the expression.