Intro to Java

Upon completion of this module, a student will be able to

- write code in a simple interactive coding environment
- output text for applications without a graphical user interface
- output text for applications with a graphical user interface
- annotate source code to allow others to better understand it
- understand and explain the role of primitive data types
- explain the difference between primitive data types and reference data types
- understand and apply core operators in code

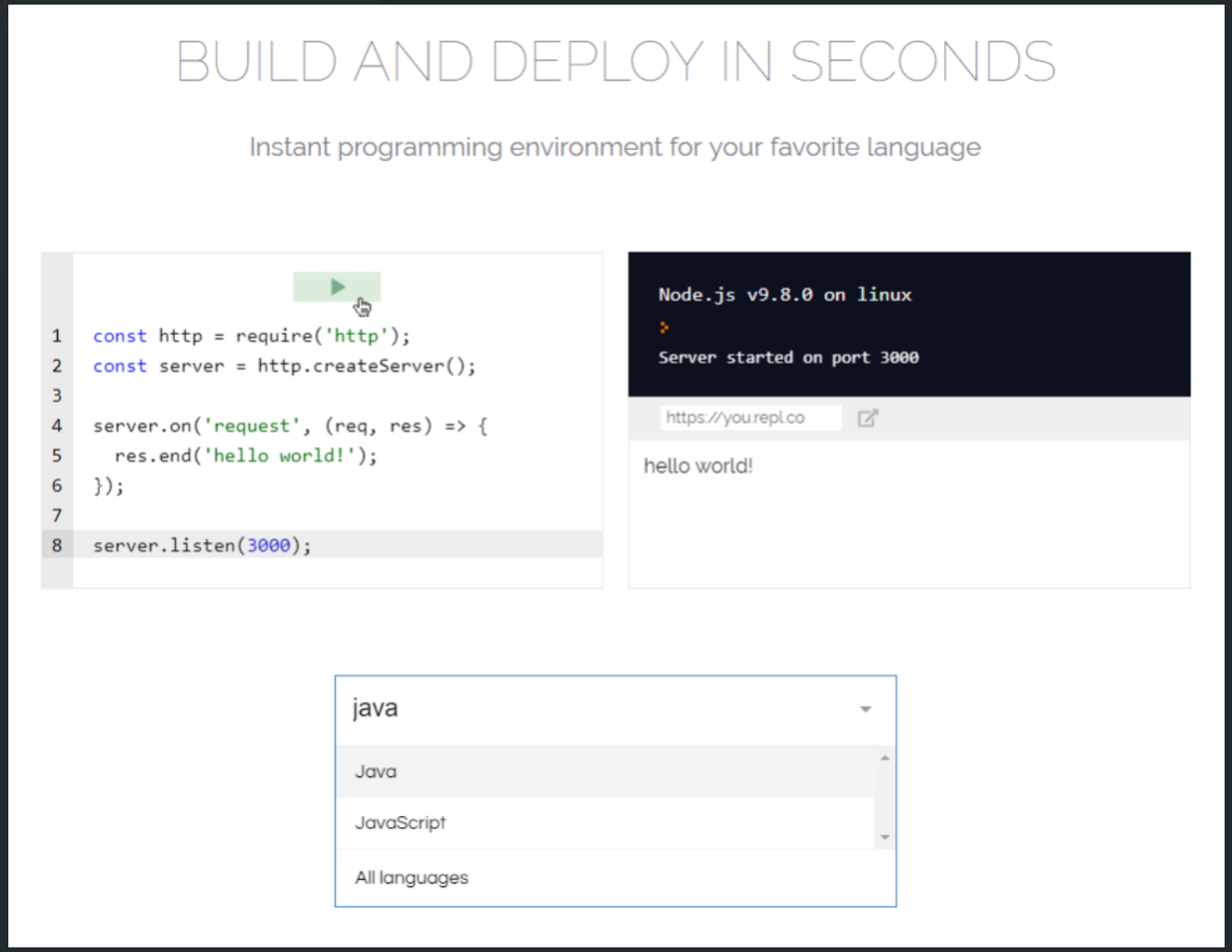
Project

- Task
  - Complete the “Intro to Java” Assignment in repl.it
- Repo
  - https://github.com/LambdaSchool/Java_Introduction
- Submission
  - Submit through repl.it
- Challenge
  - Practice performing similar tasks in android and writing the output to the UI

A Student Can

write code in a simple interactive coding environment

Select Language



A Student Can

output text for applications without a graphical user interface

Pure Java

```
  A  B C    D   E F             G
System.out.println("Hello World!");
```

A. System Object
B. Dot Operator
C. Out Object
D. Method Call
E. Parentheses to call method
F. String Parameter
G. End Semicolon

Challenge

- Write code that will write a short bio about yourself.
  - Be sure to create new lines where necessary.

- Example

```
Name: Chance Payne
Hobbies: Family, Gaming, Rock Climbing, Software Projects
Why I want to be a Developer:
I began coding on a Casio calculator in high school, writing apps and games. In college,
I started studying computer science as a stepping stone to patent law. I soon found our
that I loved writing code and really didn't want to be stuck in school any longer than necessary!
```

A Student Can

output text for applications with a graphical user interface

Android

A. Declare Data Member (Handle)
B. End all statements with semicolon
C. textView Object
D. Method Call
E. Pass String Parameter

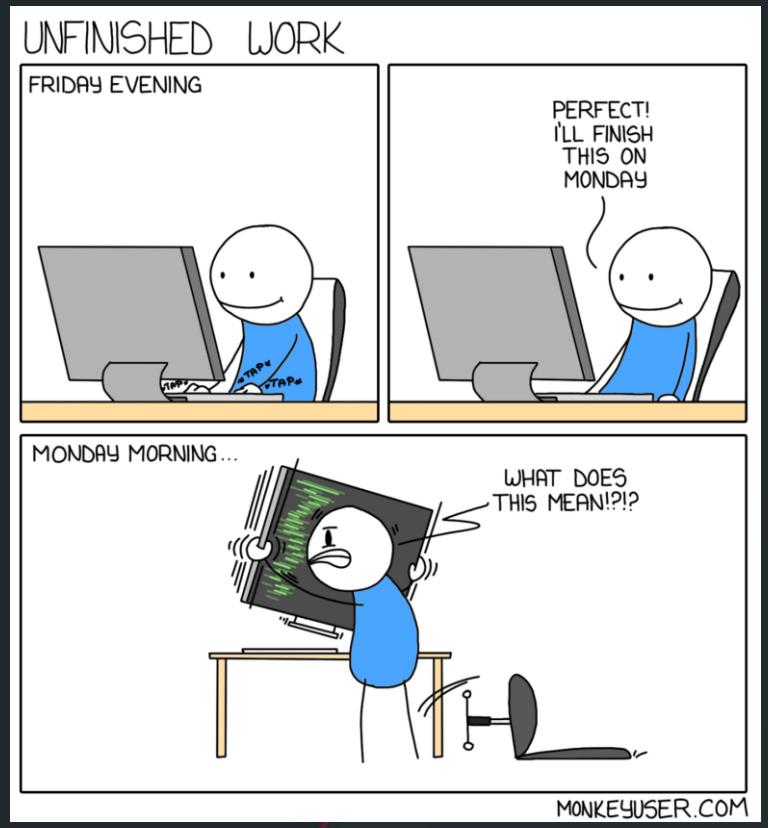
```
TextView textView;A
textView = findViewById(R.id.text_view);B
C  D      E
textView.setText("This is the new text value");
```

A Student Can

annotate source code to allow others to better understand it

Comments

- Notes in code that aren’t executed
- // single line comment
- /* comment block */
- /** JavaDoc */



A Student Can

understand and explain the role of primitive data types

Primitive Data Types

Type	Size in Bytes	Value Range	Fraction	Default Value
byte	1 byte	-128 to 127	No	0
short	2 bytes	-32,768 to 32,767	No	0
int	4 bytes	-2,147,483,648 to 2,147,483,647	No	0
long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	No	0**L**
float	4 bytes	±3.40282347E+38F (6-7 significant decimal digits)	Yes	0.0**f**
double	8 bytes	±1.79769313486231570E+308 (15 significant decimal digits)	Yes	0.0**d**
char	2 bytes	0 to 65,536 (unsigned, Unicode)	No	'**\u**0000'
boolean	*not precisely defined*	true or false	No	false

A Student Can

explain the difference between primitive data types and reference data types

Reference Data Type

- Reference data types are a combination of Primitive Data Types and Functionality
- They reference an object

A Student Can

understand and apply core operators in code

Arithmetic Operators

- Assignment Operator (=) stores the value on the right of the operator in the data member on the left of the operator
- Multiplication Operator (*) multiplies operands together
- Division Operator (/) divides the right operand from the left operand
- Addition Operator (+) adds operands together
- Subtraction Operator (-) subtracts the right operand from the left operand
- Modulo Operator (%) returns the remainder of a division operation
- Unary Increment (++) increments operand by one and stores the result
  - Post-increment (x++) increments after statement executes
  - Pre-increment (++x) increments before statement executes
- Unary Decrement (--) decrements operand by one and stores the result
  - Post-decrement (x--) decrements after statement executes
  - Pre-decrement (--x) decrements before statement executes
- Precedence: http://www.cs.bilkent.edu.tr/~guvenir/courses/CS101/op_precedence.html