Basic Programming

...

Before we start...

Let's get to know each other.

What are we going to talk about?

Quiz and exercises walk-through

Let’s write our first function!

Let’s add some logic - setting variables!

What if one function isn’t enough?

Exercises :)

What did we do before?

We talked about:

- Different languages
- Program (Application)
- Code editors
- Printing values to console (terminal)
- Working with Github

..... and many more ....

Q1. What is Java?

A. A text editor
B. Version tracking system
C. Object-oriented programming language

Q2. What is the most popular programming forum?

A. stackoverflow.com
B. inrng.com
C. oracle.com

Q3. What is not a part of a program? (find 1)

A. Decisions
B. Input
C. Output
D. Implosion
E. Math

Q4. What is Java’s compilation flow

A. source code -> byte code -> output
B. byte code -> source code -> output
C. source code -> output

Q5. Which function is an entry to Java program?

A. start
B. main
C. execute

Q6. How do we print text to the console?

A. this.print(“Hello!”);
B. (new BufferedWriter(console)).write(“Hello!);
C. System.out.println(“Hello!”);

Q7. Why do we use a version control system (find 2)?

A. To connect to Github
B. To simplify tracking changes in files
C. To collaborate with others
D. To modify Java files

Q8. What you need to do to copy a repository in Github?

A. fork
B. commit
C. pull

Q9. What you need to upload changes?

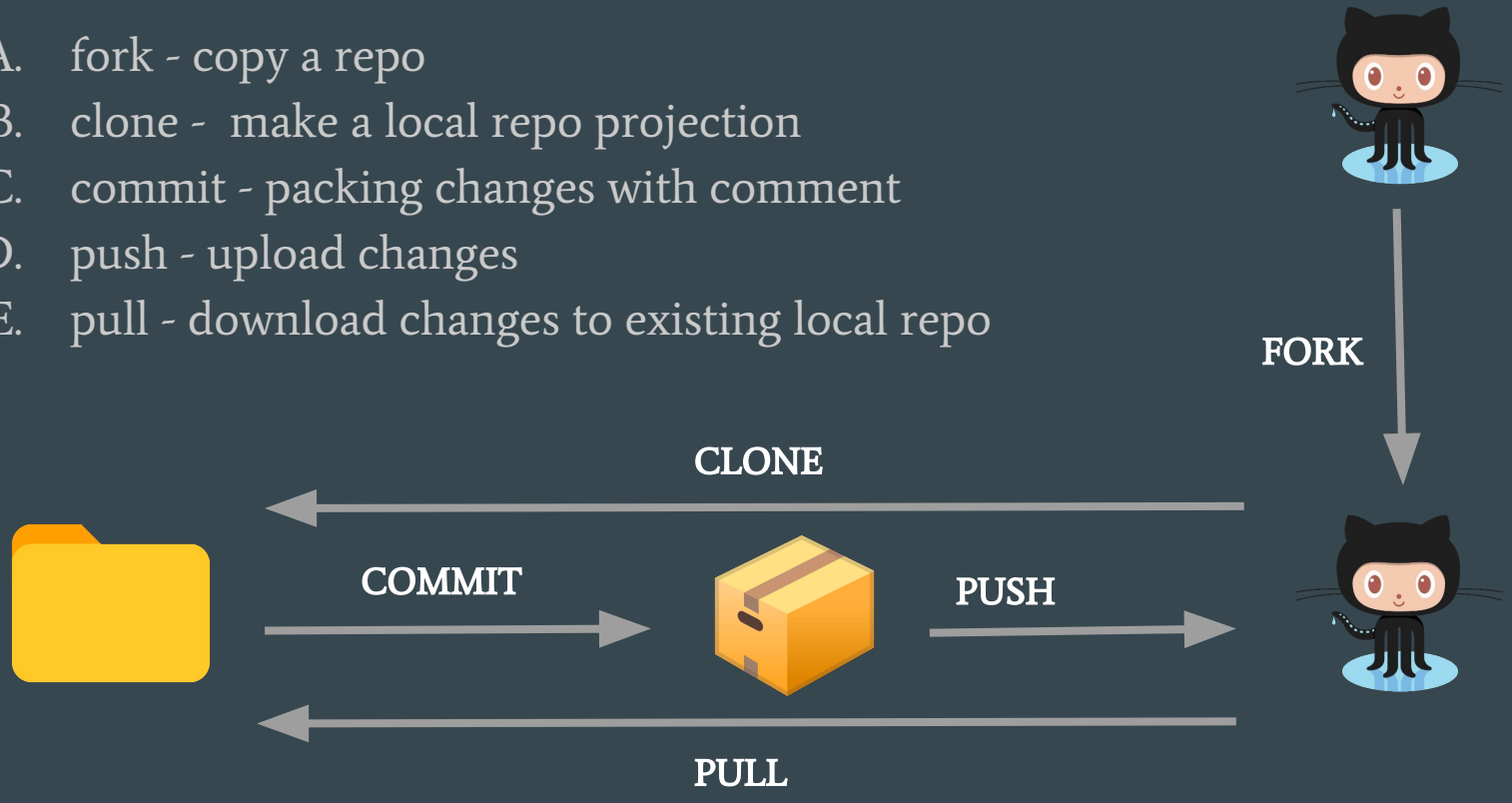
A. pull
B. clone
C. push

Q. Answers

1. C
2. A
3. D
4. A
5. B
6. C
7. B,C
8. A
9. C

Quick git revision

A. fork - copy a repo
B. clone - make a local repo projection
C. commit - packing changes with comment
D. push - upload changes
E. pull - download changes to existing local repo



Walk-through 1

- Log to Github
- Go to: https://github.com/bjowczarek/workshop-lesson-1
- Fork it. If you did it previously go to “your repositories” and delete your existing copy.
- Open VM
- Open Terminal
- Write:
  - `cd workspace`
  - `sudo rm -R workshop-lesson-1`
  - `git clone <https to your forked repo>`
  - `git config —global user.email "your email”`
  - `git config —global user.name “your username”`
- Close Terminal

Walk-through 2

- Open “Visual Studio Code”
- Open folder Home/workspace/workshop-lesson-1 - It's important as VS code will take appropriate launch.json configuration file.
- Read readme.md file
- Investigate all .java files in src/ folder.
- Go to debug and select in dropdown in upper left corner:
  - `Debug (Launch)-HelloWorldApplication<Project1Lesson1>`
  - `Debug (Launch)-MainPanel<Project2Lesson1>`
  - `Debug (Launch)-Application<Project3Lesson1>`
- Try to modify program’s output.

Why do we need different data types?

They all get encoded in binary - and it saves space to divide them up into different categories.

“1” can be just “1” if it’s an “int” (integer/liczba całkowita) + parity

“1” can be “true” if it’s a boolean

We encode fractions in binary with a “floating point” (that’s why it’s called “float”)

A float encoded as 1.1 is 1.5 in decimal (explained on the board)

The computer would get confused if you didn’t start by saying WHICH encoding you’re expecting!

Let's try it!

```
public class HelloWorld{

    public static void main(String []args){

    int a = 5;

    int b = 2;

    System.out.println(a/b);

    }

}
```

Now you can use your programming editor as a calculator!

```
public class HelloWorld{

    public static void main(String []args){

    float a = 5;

    int b = 2;

    System.out.println(a/b);

    }

}
```

adding	+
subtracting	–
multiplying	*
dividing	/
add one	++
subtract one	--

Now you can use your programming editor as a calculator!

```
public class HelloWorld{

    public static void main(String []args){

    float a = 5;

    int b = 2;

    System.out.println(a/b);

    }

}
```

adding	+
subtracting	–
multiplying	*
dividing	/
add one	++
subtract one	--

EXERCISE: TRY “%” AND TRY TO FIGURE OUT WHAT IT DOES

ASCII: how are letters typically encoded?

https://en.wikipedia.org/wiki/ASCII (English)

https://pl.wikipedia.org/wiki/ASCII (Polish)

Examples:

011 0001 means “1”

100 0011 means “C”

110 1111 means “o”

ASCII: how are letters typically encoded?

https://en.wikipedia.org/wiki/ASCII (English)

https://pl.wikipedia.org/wiki/ASCII (Polish)

Examples:

My name is “Eva” and a computer would encode it as:

100 0101

111 0110

110 0001

Different lengths of integer numbers

1. byte 8 bits (it doesn't make sense to use fewer)

   1 bit for sign/parity, 7 bits for number

   0 1 1 1 1 1 1 1 max value is **127** = **64+32+16+8+4+2+1** = $2^7 - 1$

0	1	1	1	1	1	1	1
Sign +	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	64	32	16	8	4	2	1

Create a byte this way: byte b = 1 ;

Q: What happens when we try this? byte b = 200;

Different lengths of integer numbers

1. byte 8 bits (it doesn't make sense to use fewer)
2. short 16 bits
3. **integer 32 bits (32 bits computers)**
4. long 64 bits

MAX VALUES:

byte b = 127;

short s = 32767;

int i = 2147482647;

long l = 92233720368547758O7; //will it compile?

Different lengths of integer numbers

long l = 92233720368547758**07**;

long l = 92233720368547758**07L;**

JAVA assumes that every number you write in the code is of 32 bit **int** type!

When there are too many numbers JAVA is lost

Long credit card number - will it compile?

```
Long l = 1234_5678_9012_3456L;
```

0 in front of an integer number

Try:

```
System.out.println( 12 );

System.out.println( 012 );

System.out.println( 0b12 );

System.out.println( 0x12 );
```

‘A’ is also a number

char - 16 bit number

char values: {0 - 65535 }

short -16 bit number

short values: {-32768 ; 32767}

short has a sign bit, char doesn’t

char is an 16 bit integer number that is **printed as a text!**

65 = ‘A’ **66 = ’B’** **67=’C’** **see ASCII table to find out more**

‘A’ is also a number

Which one is correct?

char myChar = ‘A’; or char myChar = 65;

See what happens:

```
System.out.println(A');

System.out.println('A'+'B');

System.out.println((char)('A'+'B'));
```

‘A’ is also a number

Which one is correct?

char myChar = ‘A’; or char myChar = 65;

See what happens:

```
System.out.println(A');

System.out.println('A'+'B');

//when JAVA works on ANY numbers it tends to convert everything to int

System.out.println((char)('A'+'B'));

//we need to tell java to change the result to char
```

Floating point number

Numbers in a format like this 0.9 2.0 3.14

float - 32 bits approximate range $\{10^{-45}; 10^{38}\}$

float = 0.2f;

double - 64 bits approximate range $\{10^{-342}; 10^{308}\}$

double = 0.2d; double = 0.2;

Is flota bigger than int?

float - 32 bits approximate range $\{10^{-45}; 10^{38}\}$

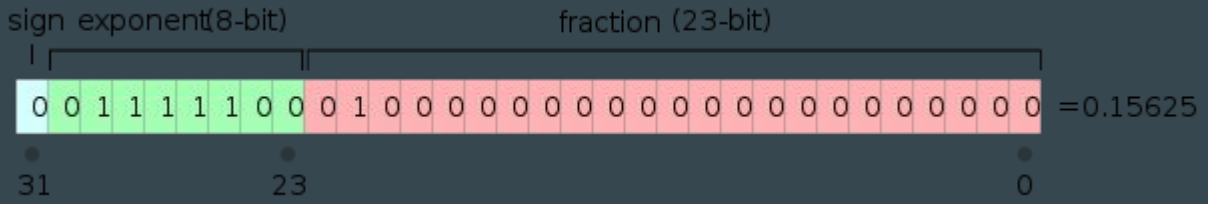
int -38 bits approximate range $\{10^{20} ; 10^{20} \}$

How come we can squeeze so many numbers into 32 bits?

We cheat! Note every number is represented we skip some **big numbers**

10^{37} and 10^{37}**+4** might be the same number to JAVA

How is float stored in a memory?



We use some mathematical **magic** https://en.wikipedia.org/wiki/IEEE_754-1985

Type casting

Sometimes we need to change the type of the variable

Java can do this automatically for us!

float f = 1f ;

int i =f;

int i =1;

float f = i;

Type casting

```
float f = 1.0f ;

int i =f;

//incompatible types: possible lossy conversion from float to int

int i =(int)f;

// You are smart, you know that the conversion makes sense. Java trusts you
```

Type casting

```
float f = 1.8f ;

int i =(int) f;

System.out.println(i);

**********

byte b = 130;

byte c = (byte)130;

System.out.println(c);
```

TYPE	DESCRIPTION	DEFAULT	SIZE	EXAMPLE LITERALS
boolean	true or false	false	1 bit	true, false
byte	twos complement integer	0	8 bits	(none)
char	unicode character	\u0000	16 bits	'a', '\u0041', '\101', '\\', '\','\n',' β'
short	twos complement integer	0	16 bits	(none)
int	twos complement integer	0	32 bits	-2, -1, 0, 1, 2
long	twos complement integer	0	64 bits	-2L, -1L, 0L, 1L, 2L
float	IEEE 754 floating point	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F
double	IEEE 754 floating point	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html

Objects vs. primitive data types

1. Primitive data type - we know how much memory space we need to write it down
   - int requires 32 bits
   - char requires 8 bits
2. Objects - we don't know how much space is required.

   String c = "Kot"; is an object of type String (3 chars = 3*8 bits)

   String c = "Kotek"; is an object of type String (5 chars = 5*8 bits)

   Objects also use memory space for methods (simple set of instructions)

   e.g. c.toUpperCase() -> uses value of c "Kotek" and changes to "KOTEK"

Boolean data type

ONLY TWO VALUES :

boolean a = false;

boolean b = true;

TRY:

System.out.println(a);

System.out.println(b+a);

System.out.println(!a);

boolean b = 0;

Some fun with Booleans (named after XIX c. logician George Boole)

You can ask your computer questions!

“Is 5 bigger than 3?” “Is 129 divisible by 3?” “Is 14*15 bigger than 10*19?”

```
public class HelloWorld{

	public static void main(String []args){

	float a = 5;

	int b = 3;

	System.out.println(5>3);

	}

}
```

If we have logic... then we can also have reasoning.

```
if(temperature<10){
System.out.println("Please wear a coat today.");
}
```

If we have logic... then we can also have reasoning.

```
int temperature = 6;

if(temperature<10){

System.out.println(“Please wear a coat today.”);

}
```

If we have logic... then we can also have reasoning.

If a is divisible by 3 and b is divisible by 3 then a*b is divisible by 6.

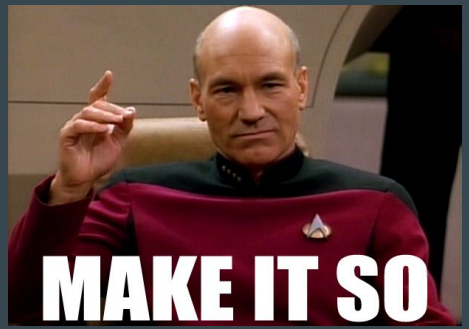
```
if(a%3==0 && b%2==0){
System.out.println(a*b/6);  }
```

&& means “and”

|| means “or”

== means “is equal to” It has to be something different from just = cause that we used to define things! For a computer a statement of fact (==) is different from an action (=).

a = 6;



vs

a == 6;

If we have logic... then we can also have reasoning.

If **a** is divisible by 3 and **b** is divisible by 3 then a*b is divisible by 6.

```
if(a%3==0 && b%2==0){

System.out.println(a*b/6); }
```

WHEN IN DOUBT WRITE IT OUT IN WORDS (WORKS FOR CODE AND MATH)

“If the remainder of **a** when divided by 3 is equal to 0 and the remainder of **b** when divided by 2 is equal to zero, print the result of dividing a times b by 6.”

|| means “or” - try coding the following:

“If the remainder of **a** when divided by 3 is equal to 0, print “a is divisible by 3”, if the remainder of **a** when divided by 3 is equal to 1 or 2, print “a is not divisible by 3”.”

```
if(a%3==0){
System.out.println(a+“ is divisible by 3”);
}
if(a%3==1 || a%3==2){
System.out.println(a+“ is not divisible by 3”);
}
```

```
if(a%3==0){
System.out.println(a+“ is divisible by 3”);
}
else{
System.out.println(a+“ is not divisible by 3”);
}
```

Exercises

Online editor:

Examples: http://tpcg.io/swjfMA

Exercises: http://tpcg.io/Y3xmfl

Github:

https://github.com/bjowczarek/workshop-lesson-2