Introduction to Java

Michelle Brush
Senior Software Architect
Cerner Corporation

New Commitments

- Time boxing
- Public questions

gdikc/IntroductionToJava | how to download from giti

https://www.google.com/search?q=how+do+download+from+github&rlz=1C5CHFA_enUS527US527&oq=how+do+download+from+github&aqs=chrome..69i57j0l5.5734j0j7&sou

Google

how to download from github

+Michelle | Share

Web | Videos | Images | Shopping | Maps | More | Search tools

About 35,300,000 results (0.38 seconds)

How to download source in .zip format from **GitHub**? - Stack Overflow
stackoverflow.com/.../how-to-**download**-source-in-zip-f... Stack Overflow
May 1, 2010 - Please Note: This is deprecated as of September 2013. See revised answer here. Have a read through http://help.**github**.com/. To clone that ...

git - Fastest way to **download** a **github** project - Stack Overflow
stackoverflow.com/.../fastest-way-to-**download**-a-**githu**... Stack Overflow
Jun 24, 2011 - When you are on a project page, you can press the '**Download** ZIP' button which is located at the bottom right. This allows you to **download** the most ...

How to download from GitHub? - Developers - Appcelerator
developer.appcelerator.com/.../how-to-**downlo**... Appcelerator Titanium
Dumb question, but I'm new to Mac and Appcelerator, and I'm having trouble downloading source from **GitHub**. Could someone please outline the basic steps to ...

GitHub for Windows
windows.**github**.com/
The easiest way to use GitHub on Windows. Download**Download GitHub** for Windows. Windows XP, Vista, 7 & 8. Dashboard. Everything you care about in a ...

How the Heck Do I Use **GitHub**? - Lifehacker
lifehacker.com/.../how-the-heck-do-i-use-**github** Lifehacker
by Adam Dachis
Feb 12, 2013 - Dear Lifehacker, I've learned to code and want to start using **GitHub** to manage ... Git makes this all happen, so you need to **download** the latest ...

GitHub For Beginners: Don't Get Scared, Get Started – ReadWrite
readwrite.com/.../understanding-**github**-a-journey-for-begin... ReadWrite
Sep 30, 2013 - But when you access their **GitHub** accounts, you're free to **download**, study, and build upon anything they add to the network. So what are you ...

A gentle guide to Git and **GitHub** | Quick2Wire
quick2wire.com/articles/a-gentle-guide-to-git-and-**github**/
You might **download** an archive (typically a zip or tgz file). You'll usually need ...

//github.com/gdikc/IntroductionToJava

ster IntroductionToJava /

2 days ago latest commit aefc8b4d0b

add pdf version of slides	2 days ago
adding class2 activity	2 days ago
add license to readme	3 days ago

luctionToJava

level course on Java with activities

://creativecommons.org/licenses/by/4.0/deed.en_US Creative Commons Attribution 4.0

icense

Issues 1

Pull Requests 0

Wiki

Pulse

Graphs

Network

Settings

SSH clone URL

git@github.com:gdik

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop

Download ZIP

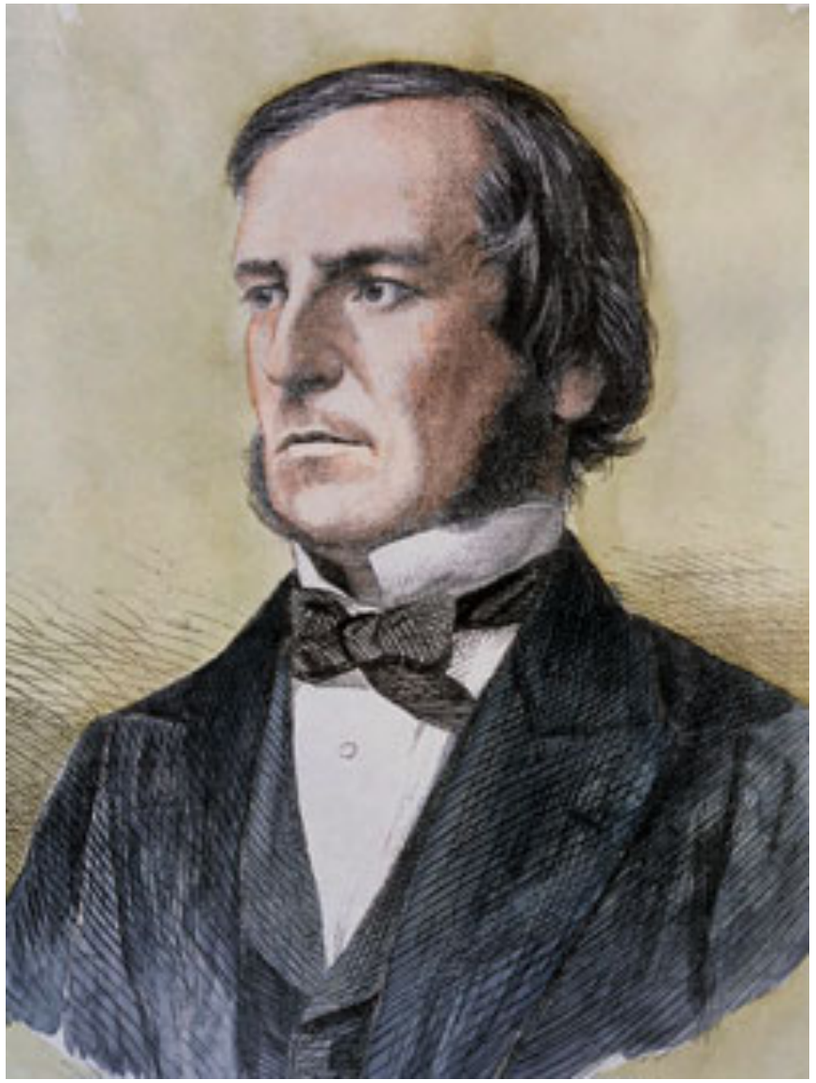
Class Structure

- The Why and What of Java
- **Basic Logic**
- Object-Oriented Programming
- When Things Go Wrong
- Common Java APIs
- JUnit
- Generics
- Putting It All Together I & II

Boolean Algebra

George Boole

What would algebra look like if there were only two possible values: true and false?



Operations

And

true **And** true is true.
true **And** false is false.
false **And** false is false.

Or

true **Or** true is true.
true **Or** false is true.
false **Or** false is false.

Not

Not true is false.
Not false is true.

English:
Pizza or Hot Dog?

You can only have one.

Computers:
Pizza or Hot Dog?

You can pig out.

Not true **Or** false

true **And Not** false

Not Not false

Java Operations

And

&&

Or

||

Not

!

http://www.javarepl.com/console.html

```
! true || false

true && ! false

! ! false
```

What's the point of all this?

Logic & Control

A lot of programming involves **conditional statements** and **loops**.

Conditional Statements

```
if ( some condition ) {
    //do something here
} else if (some other condition ) {
    //do something different here
} else {
    //do something completely different
}
```

Conditional Statements

What's this?

```
if ( some condition ) {
    //do something here
} else if (some other condition ) {
    //do something different here
} else {
    //do something completely different
}
```

Comments

// Nothing after the slashes counts

/* Nothing in between counts */

Comments

/* Comments allow you to remind
yourself and future engineers
what on earth you were thinking */

Conditional Statements

```
if ( age < 13 ) {
    return Menu.KIDS;
} else if (age > 64 ) {
    return Menu.SENIORS;
} else {
    return Menu.STANDARD;
}
```

Conditional Statements

```
if ( age < 13 ) {
    return Menu.KIDS;
} else if (age > 64 ) {
    return Menu.SENIORS;
} else {
    return Menu.STANDARD;
}
```

Conditional Statements

What is this?

```
int age = 21;
if ( age < 13 ) {
    return Menu.KIDS;
} else if (age > 64 ) {
    return Menu.SENIORS;
} else {
    return Menu.STANDARD;
}
```

Variables

variable

```
int age = 21;
if ( age < 13 ) {
    return Menu.KIDS;
} else if (age > 64 ) {
    return Menu.SENIORS;
} else {
    return Menu.STANDARD;
}
```

Variables

I have an integer called 'age' that is equal to 21.

```
int age = 21;
if ( age < 13 ) {
    return Menu.KIDS;
} else if (age > 64 ) {
    return Menu.SENIORS;
} else {
    return Menu.STANDARD;
}
```

Variables

Is an integer an object?

Does it have a class?

```
int age = 21;
if ( age < 13 ) {
    return Menu.KIDS;
} else if (age > 64 ) {
    return Menu.SENIORS;
} else {
    return Menu.STANDARD;
}
```

Objects vs. Primitives

Primitives	**Objects**
int	everything else
long	
short	
float	
double	
boolean	
char	
byte	

Objects vs. Primitives

Primitives

int
long
short
float
double
boolean
char
byte

Objects

everything else

There are things you can do with each that you can't do with the other.

Objects vs. Primitives

Primitives

int
long
short
float
double
boolean
char
byte

Objects

everything else

You can use **operators** on primitives.

Objects vs. Primitives

Primitives

int
long
short
float
double
boolean
char
byte

whole numbers

Objects

everything else

Objects vs. Primitives

Primitives		**Objects**
int		everything else
long		
short		
float	decimal	
double	numbers	
boolean		
char		
byte		

Objects vs. Primitives

Primitives		**Objects**
int		everything else
long		
short		
float		
double		
boolean	true or false	
char		
byte		

Objects vs. Primitives

Primitives		**Objects**
int		everything else
long		
short		
float		
double		
boolean		
char	characters	
byte		

Objects vs. Primitives

Primitives	**Objects**
int	everything else
long	
short	
float	
double	
boolean	
char	
byte raw data	

Bytes

8-bits of raw information

00000000
11111111
01010101
01110101

$2^8-1=255$

Conditional Statements

```
if ( age < 13 ) {
    return Menu.KIDS;
} else if (age > 64 ) {
    return Menu.SENIORS;
} else {
    return Menu.STANDARD;
}
```

What is this?

Conditional Statements

```
if ( age < 13 ) {
    return Menu.KIDS;
} else if (age > 64 ) {
    return Menu.SENIORS;
} else {
    return Menu.STANDARD;
}
```

What is this?

Enumeration

```
enum Menu {
    KIDS,
    SENIORS,
    STANDARD
}
```

a fixed list of things

predefined constants

Conditional Statements

```
if ( age < 13 ) {
    return Menu.KIDS;
} else {
    return Menu.STANDARD;
}
```

Conditional Statements

```
if ( age < 13 ) {
    return Menu.KIDS;
}
return Menu.STANDARD;
```

Putting It Together

Rules:

- You can only get a happy meal if you are a kid.
- A kid is someone 12 or under.
- If you are an adult, you get the standard meal.
- If you are a boy kid, you get the meal with the boy toy.
- If you are a girl kid, you get the meal with the girl toy.

Enumerations

```
enum Meal {
    GIRLS_HAPPYMEAL,
    BOYS_HAPPYMEAL,
    ADULT_MEAL
}
```

```
enum Gender {
    FEMALE,
    MALE
}
```

Putting It Together

```
if ( age < 13 && gender == Gender.FEMALE ) {
    return Meal.GIRLS_HAPPYMEAL;
} else if ( age < 13 ) {
    return Meal.BOYS_HAPPYMEAL;
} else {
    return Meal.ADULT_MEAL;
}
```

Putting It Together

```
if ( age < 13 ) {
    if( gender == 'F' ) {
        return Meal.GIRLS_MEAL;
    } else {
        return Meal.BOYS_MEAL;
    }
} else {
    return Meal.ADULT_MEAL;
}
```

Switch

```
if ( age < 13 ) {
    switch(gender) {
        case Gender.FEMALE:
            return Meal.GIRLS_MEAL;
        case Gender.MALE:
            return Meal.BOYS_MEAL;
        default:
            return Meal.BOYS_MEAL;
    }
} else {
    return Meal.ADULT_MEAL;
}
```

Switch

```
if ( age < 13 ) {
    switch(gender) {
        case Gender.FEMALE:
            return Meal.GIRLS_MEAL;
        case Gender.MALE:
        default:
            return Meal.BOYS_MEAL;
    }
} else {
    return Meal.ADULT_MEAL;
}
```

Loops

What if we need to do this for every person in the party?

for

```
for (operation; condition; operation) {
    //some work
}
```

for

```
for (int i = 0; i < 5; ++i) {
    System.out.println("Welcome!");
}
```

for

What is int?

```
for (int i = 0; i < 5; ++i) {
    System.out.println("Welcome!");
}
```

for

Why 0?
Why not 1?

```
for (int i = 0; i < 5; ++i) {
    System.out.println("Welcome!");
}
```

for

What is this?

```
for (int i = 0; i < 5; ++i) {
    System.out.println("Welcome!");
}
```

++i and i++

i++ and **++i** both mean **add 1 to i**.

i++ means do it ***after*** this line of code.

++i means do it ***before*** this line of code.

```
int i = 0;
if(i++ == 0)
```

```
int i = 0;
if(++i == 0)
```

(operators)

http://www.javarepl.com/console.html

for

For each number counting from 0 to 4, say, “Welcome!”

```
for (int i = 0; i < 5; ++i) {
    System.out.println(“Welcome!”);
}
```

for

For each number counting from 0 to 4, print it!

```
for (int i = 0; i < 5; ++i) {
    System.out.println(i);
}
```

for

I'm on the bottom floor.
I need to get to floor 9.
I need to go up the stairs.

```
for (int floor = 0; floor < 9; ++floor) {
    takeStairs(floor);
}
```

European Buildings

Useless Trivia:

In Europe, the first floor is 0.

The second floor is 1.

for

You can loop over all values in an enumeration.

```
for (Meal meal : Meal.values()) {
    System.out.println(meal);
}
```

for

"For each meal in the collection of meals..."

```
for (Meal meal : Meal.values()) {
    System.out.println(meal);
}
```

do

```
do {
    //some operation;
} while (//some condition)
```

do

```
do {
    pourGas(tank);
} while (!tank.full())
```

Do fill the gas tank while it's not full.

while

```
while (//some condition) {
    //some operation;
}
```

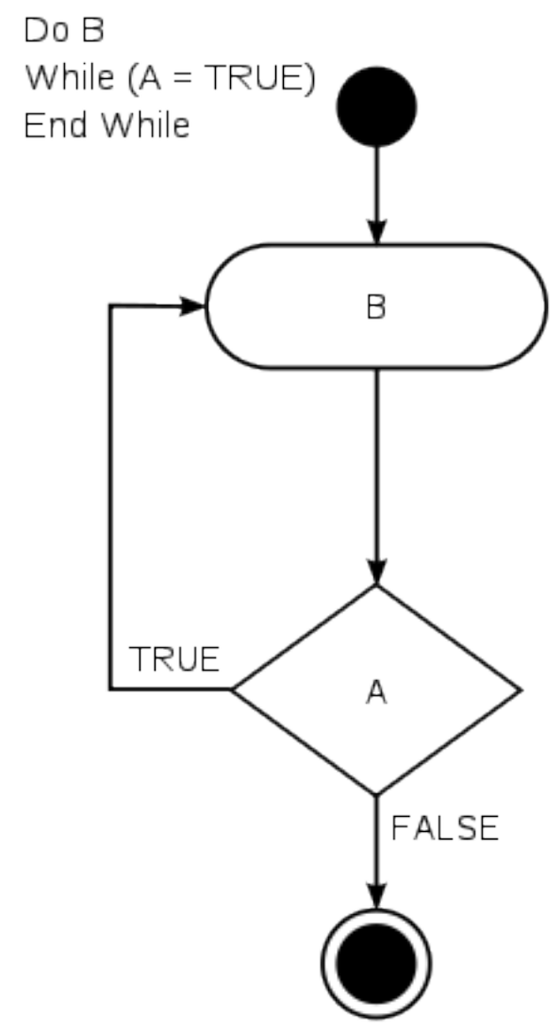
while

While my coffee isn't sweet, add a sugar.

```
while (!coffee.isSweet()) {
    coffee.addSugar(1);
}
```

Loop Activity

Which loop should you use?



I'm a barista at a coffee shop. My job is to make the coffee based on what cups are labeled in front of me. I can't stop until there are no more cups.

- for
- do while
- while

I'm a barista at a coffee shop. My job is to make the coffee based on what cups are labeled in front of me. I can't stop until there are no more cups.

- for
- do while
- **while**

```
while(queue.hasCup()) {
    Cup cup = queue.nextCup()
    brewCoffee(cup);
}
```

I'm at the gym. I have do 20 leg lifts.

- for
- do while
- while

I'm at the gym. I have do 20 leg lifts.

- **for**
- do while
- while

```
for(int i = 0; i < 20 ++i) {
    leg.lift(120);
}
```

There's a new season on Netflix. I want to keep watching episodes until I reach the end of the season.

- for
- **do while**
- while

```
do {
    episode = season.getNext();
    watch(episode);
} while(season.hasNext());
```

Making Change

- You will take the number of cents as input.
- You have quarters, dimes, nickels, and pennies.
- You must make change.
- Print out each coin dispensed.

Input: 87
Output:

```
quarter
quarter
quarter
dime
penny
penny
```

Making Change

- You will take the number of cents as input.
- You have quarters, dimes, nickels, and pennies.
- You must make change.
- Print out each coin dispensed.

Input: 87
Output:

```
quarter
quarter
quarter
dime
penny
penny
```

https://github.com/gdikc/IntroductionToJava/blob/master/class2/samples/change/ChangeMachine.java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class ChangeMachine {

    public static void main(String[] args) throws IOException {

        System.out.print("Enter an amount of money:");

        BufferedReader bufferedReader = new BufferedReader(new
                                InputStreamReader(System.in));
        int cents = Integer.valueOf(bufferedReader.readLine());
        System.out.println("You entered " + cents + " cents.");

        //TODO: Make Change!

    }
}
```

Input: 26
Output:

```
quarter
penny
```