Revature Notes

Nathan Christopher Medrano

* 6/17/19

Trainer: William Ona

Email: [william.ona@revature.com](mailto:william.ona@revature.com)

Getting Started:

Friday – Travel reimbursement given on payment card. “Friday with HR”

* + Time sheet, set up direct deposit, etc.
  + Lunch provided that day (Find food places)
  + HR talk and help

Parking – Vehicle information already passed on,

* + You will not get ticketed or cited
  + Park in Faculty/Staff Lots
  + Shuttle? Download Doublemap app
  + Rideshare? VIA app

Housing – Will be going to Liv+ soon enough

* + Will go as group

Meetings today:

1st William Gentry – Technology and Delivery Team

* + What type of company is Revature? Consultant, Technology Solutions, Talent Development
    - Entry level for Software-Engineers
  + Check news for Revature

2nd Steven Kelsey – Technology Manager at USF Location

* + Focus on Technology
  + Help each other
  + Curriculum changes in accordance to Company desire
  + What we’ll learn:
    - Programming Skills, Enterprise Project Development, Soft Skills and Career Coaching
  + Bootcamp Program Structure:
    - Build Foundations
      * OOPs, Java, DB
      * Environment & Tools
      * Web App
      * Project 0
    - Software Architecture
      * Design Patterns
      * N-Tier Architecture
      * MVC
      * Project 1
    - Adopt DevOps
      * Agile/Scrum
      * DevOps Tools
      * Testing
    - Advanced Concepts
      * Cucumber & TestNG
      * Selenium
      * Protractor
      * Project 2
    - Work on a Live Application
      * Two Weeks Iteration with full SCRUM
      * Enterprise best practices
      * Project Showcase
      * Project 3\*
        + Whole Group working on whole project
        + Group Presentation
        + Show honed skills (both interviewing and programming)
    - Monitor Progress
      * Weekly Assessment – every Monday
      * QC (quality control) Process (Weekly quizzes, pop)
        + Any complaints or problems with training can be addressed here
        + You will be told you are being evaluated and be constantly updated
      * Expert Panel
        + Will always be done nonetheless
        + Even after Client hires you
  + Capabilities:
    - Will be taking the Oracle Examination Exam, main goal to keep up with
      * Payment for exam will be reimbursed
      * When: William Will hand out practice exams this or next week
        + Between weeks 6-10 will be consideration for actual test
        + Or before going to Client
      * Highly recommended, will help to build portfolio and resume
      * $10 for practice exams (also reimbursed), enthaware tests
      * Passing is 65%
    - Projects will have been made by a previous group, with our group’s intent to fix it
    - Keep yourself disciplined
  + Company Points of Contact
    - Trainer: William Ona
      * email: [william.ona@revature.com](mailto:william.ona@revature.com)
    - Recruitment: Cameron Coley
      * email: [Cameron.coley@revature.com](mailto:Cameron.coley@revature.com)
      * or email: [michael.minton@revature.com](mailto:michael.minton@revature.com)
    - Housing:
      * [housing@revature.com](mailto:housing@revature.com) – supplies, check-in, check-out
    - Human Resources (HR):
      * [hr@revature.com](mailto:hr@revature.com)
    - If seek or need to be absent, let William Ona know asap
    - Even in regards to projects, let William know
  + Set-up Time
    - Slack (Discord for businesses)
  + Schedule:
    - Week 1: Java
    - Week 2: SQL
    - Week 3: Front-End Tech (CSS, HTML, JS),
      * Project 0 Due\*\*\*\*\*\*\* (Simple Console Application)
        + Individual
    - Week 4: Angular(abstraction framework), Servlets
    - Week 5: DevOps/Intro to Testing
      * Automation, DevOps Pipeline
      * Testing Frameworks
      * Project 1 Due\*\*\*\*\*\*\*\*\*\* (Full Stack Application)
        + Group
    - Week 6: Testing Automation/ Intro to Spring(a series of abstraction frameworks to code more easily)
    - Week 7: Advanced Spring/ Microservices
      * Project 2 Due\*\*\*\*\*\*\*\*\*\*
    - Week 8-10: Project 3 to be worked on
      * Portfolios
      * Panels
      * Potential Interviews (Maybe)\*
        + This happens depending on when client is ready
  + Evaluations:
    - Every week – Online Quizzes (Material of Previous Week)
      * One-on-one Interviews
      * QC (Quality Control) – Someone other than William quizzes us
  + Git Commands
    - git config –global user.name
    - ls – check file contents
    - branch adding:
      * git checkout –b NathanMedrano
    - touch – to create file
      * touch FileName.txt
    - Edit .txt? vim file.txt
      * in vim: esc, i - to edit
      * :wq (w to write, q to close, colon needed), or :x
      * :q! (force quit)
      * committing (saying yes this is valid version of application)
        + git status

checks for branches that are eligible to be added

* + - * + git add filename.txt
        + git commit –m “”
    - Note: this is all locally at first
      * to get remote to local, git pull
        + git pull origin master
      * local to remote, git push
        + git push origin nathanmedrano
      * Note there are many other commands and many repositories
  + Java Time
    - OOP, Object-Oriented Programming Language
      * Using programming to create ‘entities’ in order to fulfill a goal
      * Not fully OOP, due to primitives
    - Cool things:
      * Not in hands of developer to manage memory as the Developer
      * Highly portable, “write it once, run it anywhere!”
        + Doesn’t matter the operating system you are you using
      * Compiling Language, translate your code into something your machine can execute
    - Terminology:
      * IDE – Integrated Development Environment
        + Software used to write code (usually comes with other tools and can compile or run your code)
      * Text Editor – Editor that is just for editing text and code
      * STS (Spring Tools Suite) – an IDE that supports Java
        + Based on Eclipse
        + Comes with specific plug ins that we need for development
        + Works well with Spring
    - Naming
      * Class names: ClassName
      * Variables: varName
    - You can only have one public class
* 6/18/19
  + Recall: object = an instance of a class that Java uses to take action
    - class = describes state and behavior, a blueprint
    - state = represented by variables or fields of class
    - behavior = representation by methods
  + **JVM, JRE, JDK**
    - JVM = Java Virtual Machine
      * Recall process of compiling, JVM is the actual structure that executes the computer code
      * When we compile .java to .class, we convert into machine readable java byte code
      * The JVM is the structure that will execute this bytecode
        + This is what makes Java portable! All you need is the JVM that can execute these instructions
      * When we call ‘java <file name>’ in command line, we are telling the JVM to execute the instructions specified in that file
    - JRE = Java Runtime Environment
      * This is the only thing you need to run java files; needed to run java (think access to certain things),
      * Contains the JVM
      * The JRE consists of the JVM and supporting libraries needed to aide JVM
    - JDK = Java Development Kit
      * Software development environment used to develop java applications
      * Contains JRE and development tools (Compiler, Archiver, etc.)
      * When calling ‘*javac*’ you are using the compiler to compile .java files
      * Includes Dev Tools
    - JDK > JRE > JVM
  + Use ctrl + shift + f to auto format
  + In java, printing an empty variable prints default 0 case
    - if object, will print ‘null’
    - /\*
    - Variable casting is the conversion of a varibale's datatype to another. Casting comes in two flavors:
      * 1) implicit casting (widening)
        + You do not have to actually write out the cast
      * 2) explicit casting (narrowing)
    - \* - You DO have to write out the cast : (newDatatype) varName
    - \*
    - \* When it comes to primitive casting, you can mostly perform implicit casting
    - \* if you are converting a smaller datatype to a bigger one.
    - \* However, when converting a larger datatype to a smaller one, explicitly
    - \* stating that you intend to do so is required.
    - \*/
  + Julie Seals Meeting: Staging manager
    - Staging = what comes after training
    - Once all things are figured out with the client, you get to work from home at first.
    - Average of Virtual Staging, 2-3 weeks
  + Review questions for QC next Monday\*\*\*\*\*\*\*\*
  + Summary of today: Access Modifier, Non-Access Modifiers, Primitives, Scanners, Control Statements
* 6/19/2019:
  + BY MONDAY!
    - **Make an Amazon Web Service Account** - FINISHED
    - **Download Entheware for OCA and take 1 practice test and send William the score**
  + Memory in Java
    - Stack and Heap: two main structures in Java regarding memory
      * Data is stored in either thing
    - Stack: think of stack of items (stacked boxes)
      * Storage of:
        + Local variables
        + Reference variables
        + Method indications
      * Stacks follow: Last In, First Off
      * Stack Overflow: You ran out of memory
      * Threading: each thread gets its own stack!
    - Heap: think more amorphous object
      * This is where all the objects live
      * Instance variables go here
      * String Pool:
        + Place in heap allocated for Strings
        + Recall Strings are immutable; they cannot be changed
        + However, they can be concatenated
    - Reference v. Local variables:
      * ex| Object o = new Object();
      * ‘o’ is the reference variable to be stored in the stack
      * with ‘new Object()’, the instance variable is made
      * Note: main method will be added in beginning
  + Exceptions:
    - Anytime an app behaves in a way that wasn’t expected of it that can be “reasonably” recovered from (errors cannot be reasonably recovered from)
    - Exceptions come from “Throwable”
    - Exceptions you can “handle” to bypass
    - Throwable:
      * Exceptions
        + Runtime Exception

Null Pointer: calling a method on an object that has the value of Null

Arithmetic: Doing an impossible mathematical operation (1/0);

Class Cast: When you try to cast on an object that is not allowed

* + - * + IO Exception

To be done tomorrow

* + - * + File Not Found
      * Errors
    - Two main Groupings of Exceptions: Checked vs Unchecked Exceptions
      * Checked: Exceptions that are checked at compile-time.
        + If some code within a method throws a checked exception, the method must either handle the exception or must request that the exception is handled later.
        + Includes:

Everything not runtime

* + - * Unchecked: exceptions that are not checked at compile-time.
        + You do not have to handle them…but you still should
        + Includes:

all Runtime Exceptions

* + Pillars of OOP: Inheritance, Polymorphism, Encapsulation, Abstraction
    - Encapsulation: using access modifiers to restrict modification of data throughout program.
      * Uses getters and setters to interact with resources
      * Often thought of as data binding: the binding of data together in a class
      * Supplemental to the OOP paradigm because all of the information about an object is in one place
    - Abstraction: reducing code complexity and redundancy providing only needed code for use while hiding how it works.
    - Inheritance: a class’s ability to inherit methods and fields from a parent class to a child class.
    - Polymorphism: giving multiple forms and uses to an object
      * Overriding : inherit from parent and change specifically for child class
      * Overloading: having the same method signature, but having different executions
      * Covariance: If A a = new B(); where B is child of A
* 6/20/2019
  + **ACTIVATE CARDS**
  + Meeting today: QC Orientation with Kevin Tran Huu
    - Be good during interview and demonstrate soft skills
  + **Garbage Collection**:
    - Java has its own garbage collector, unlike other languages in which you must handle unused objects yourself to free up memory
    - The Garbage Collector: an execution that runs along-side a java application and is typically reclaiming memory from destroyed objects
      * An object is a candidate for garbage collection if its reference points to null or the scope in which it was created has passed
      * You cannot force the garbage collector to actually initiate the collection process; java gets to it eventually
    - When destroying an object, the collector calls that object’s finalize method
      * Every object has a finalize method, but it is empty
* 6/21/2019
  + Collection API
    - Collection: an interface referring to a group of bleh
    - Iterable [I] < -- (inherited) Collection [I]
    - From Collection:
      * Set [I] – Do support random access, but do not allow duplicates
        + From set:
        + HashSet [C]
        + SortedSet [I]

TreeSet [I]

* + - * Queue [I] – Data Structures supports FIFO, LILO; they are only processed in the order that they are added and do not support random access
        + Priority Queue [C]
      * List [I] – Supports random access (you can access any random index), allows duplicates
        + ArrayList<> [C]

Most common implementation of lists; increases with size when capacity is reached

[1,23,4,4,333,42]

myArrayList.append(3);

* + - * + Linked List [C]

Don’t have random access

They use nodes that point to the location of other nodes (before and after)

Easy to add items to the list

o-o-o-o

* + - * + Vector [C]

Stack [C]

* + - Map [I] – Note: not part of the tree (does not inherit from anything), is a part of collection API
      * Hash Table [C]\*
      * Hash Map [C]\*
      * Sorted Map [C]\*
    - Object [C]
      * Array [C]\*
        + An Array is NOT a Collection
      * Collections [C]\* (different from collection)
        + Collection interface != Collections class
        + Has methods that work for collections
  + Meeting today: David Fay, HR Orientation
    - Contacting HR: email – [hr@revature.com](mailto:hr@revature.com)
      * Payroll and benefits
      * Expense reimbursements
      * Time entry issues
      * Policy Questions
    - Housing team email:
      * [UTAHousing@Revature.com](mailto:UTAHousing@Revature.com)
        + Use this for if Cameron is not available
    - Payroll:
      * Pay in arrears, expect first pay check 5th of July
      * Direct Deposit will be set up later (first couple checks will be paper)
    - Time Entry
      * Use Revature portal to do so (William will remind us each Friday)
      * Record use of PTO
      * Needs to be in 6pm Eastern Time (5pm Central)
    - Hours Worked
      * Hours spent in classroom
      * Hours spent in outside meetings
      * Self-Study does not count!
      * Should not exceed 40 hours/week
    - Holidays!
      * July 4th – still put in your 8 hours
    - PTO – Paid Time Off
      * Accrues after 6 months of employment
        + in ADP payroll system
        + Earned at rate of 5.54 hours/ pay period
        + Caps at 200 hours\*\*\*\*\*\*\*\*\*\*\*\*
    - Expenses: Expense Reimbursement
      * Travel more than 50 miles = $250
      * Going on flights gets an extra $250, max total reimbursement of $500
      * These are done on paycard
    - Benefits:
      * Medical: Anethem/ 60-day Waiting Period
        + 2 plans available
      * Dental: Guardian / 60-day Waiting Period
        + 2 Plans
        + Standard , “Higher-Cost”
      * Vision: EyeMed / 60-Day Waiting Period
        + One Plan
    - Medical Insurance:
      * Two plans:
        + HAS Plan: Low Premium/high deductible
        + PPO Plan: Higher Premium/ Lower Deductible
      * Regardless, get LiveHealth Online
        + Telehealth option provided (video chat diagnoses)
        + Free sign up
        + Psychiatric help
      * EPA – Employee Assistance Program (EAP)
        + Health, Family, and Financial help
    - Life Insurance and AD&D
      * $50,000 Coverage for Each, Company Paid!
      * Basically it comes with it
    - Commuter Benefit
      * Contribute tax-free deductions for mass transit or parking expenses
    - Long-Term Disability
      * 90-day Waiting period
    - 401k Retirement Savings
      * Enroll through ADP after first payroll
      * No company match at this time\*\*\*\*\*\*\*\*\*\*
    - Employee Handbook
      * Key Policies Below:
        + No drugs ):
        + Standard Operating Hours/Attendance
        + Safety
        + Dress Code
        + etc.
    - Next Steps: Time and Expense Portal
    - ADP
  + Offer-Day Document Review: Michael Minton
    - Offer-Day is next Wednesday, 26th
    - Today we will receive 4 documents that will be signed through DocuSign
    - When move, don’t lock yourself into a lease
    - Odds of relocation are very low
    - Will get 500 to move
    - Review all 4 documents before Wednesday
  + Threads:
    - Thread: a line of execution (the main thread is a good example)
    - Multi-Threading: the scenario where multiple lines are executing “concurrently”
      * doing so may increase efficiency immensely
      * Very useful in multi-user scenarios
      * Accomplished through time-slicing
        + main thread - >
        + thread I - >
        + thread II ->
      * The threads take turn
    - 6 States recognized by the JVM
      * New: thread created
      * Runnable: when start method has been invoked
      * Waiting: where you have told the thread to wait until another thread wakes it up
      * Timed-Waiting: Waiting is invoked for some x amount of time
      * Blocked: thread is not able to finish its task
      * Terminated: Thread is done executing
    - New -> Runnable -> Blocked or Terminated or Waiting
    - Race Condition: the result of a transaction is dependent on the order threads access it
      * If you use synchronization, you can limit access to a resource to only one thread at a time (the keyword)
    - Blocked
      * Deadlock: When two or more threads will not give up their own resource until they get each other’s resource
      * Livelock: When two or more threads will not take another’s resource until their resource is taken
      * Thread Starvation: When a thread cannot accomplish its task (as a result of situations like livelock or deadlock)
    - Producer-Consumer Problem
      * Producer: Job to generate data and put it in a buffer
      * Consumer: must remove data from a buffer
      * Problem: make sure that producer won’t try to add data into buffer if it is full, and consumer should not consume if buffer is empty
      * Solution? Producer must sleep if buffer is full, consumer must sleep if buffer is empty
      * They will each wake up the other when it is appropriate for the other to continue
  + Comparable vs Comparator
    - Interface called “Comparable”
      * Requires you to provide implementation for the compareTo() method
      * compareTo() will compare itself to another object that you pass in as an argument
      * Used for sorting; up to dev
      * natural order
    - Comparator:
      * any third party objects that compare two other objects to each other
      * Does not follow natural order
      * Associated to compare() method