Trainers: Nick Escalona(Main Trainer), Fred

fred@revature.com

nick.escalona@revature.com slack message best way to reach

sadiki sarkar?

Julie seals staging manager
julieseals@revature.com

10 weeks training
2 weeks staging: interviewing/client meets
12 weeks total

Patrick walsh-manager

Training topics
c#, asp.net,
ado.net,
javascript, angular
sql, database,
wcf,rest, soap
aws, azure, windows server
devops:jira, Jenkins, maven, sonarqube, github, docker, kubernetes

project 0,1,2,3

10 week overview

1. C#, OOP
2. SQL, Entity Framework
3. ASP.NET MVC, HTML, CSS (Present project 0)
4. Devops, Azure, Docker (Project 1 Due)
5. Web services, REST, SOAP, ASP.NET WCF, Javascript
6. Angular
7. Microservices, Kubernetes, Docker (Project 2 Due)
8. Project 3
9. Project 3
10. Project 3

Daily work
Lunch 12:30-1:30
Moday – evaluations, quiz on last week,sometimes code assignments (timed), one-on-one mock interview, project work time
Tuesday – Training/content
Wednesday – Training/content
Thursday – Training/content
Friday – leftover content/review, project work time

SLACK DELETES MESSAGES AFTER A FEW DAYS!!! STORE IMPORTANT STUFF ELSEWHERE!!

Shell basics:

Prompt: currently logged-in user, and computer name
In yellow: current directory
You are always in one directory at a time, given by the yellow indicator
~is a special name for your home/user directory
Cd ".." : the parent of the current directory

Shell (bash) commands:

Cd: change directory to whatever dir you give as an argument
Ls: list contents of current directory

Git commands:

Git clone <.git url>

Creates a new local repository from GitHub

Git pull

To pull all new updates from GitHub since the last time you pulled

Git push

Push new updates to GitHub(first need to add files and/or commit updates to files to use push)

February 4/23/19

- Anatomy of code – language compiler, runtime, platform
- Environment setup-IDE, editor, version control
- Basic topics – core C#, program structure, testing, logging
- .NET building blocks- Framework, Standard, Core, project, solution, assembly, library, application
- Common language runtime – BCL, CIL, CLI, CLR, CTS, JIT, VES
- Runtime environment – garbage collection, managed, unmanaged
- Data types – reference, value,
- Access modifiers – internal, private, protected, public
- Extended modifiers – abstract, const, new, override, partial, readonly, sealed, static, virtual,
- Class – constructor, field, method, property, reference type
- Struct – value type
- Interface
- Enum
- Semantic code – DRY, inline/XML, comments, separation of concerns, KISS
- Object oriented programming – abstraction, encapsulation, polymorphism, inheritance
- Working with types – casting, as, boxing, is, out, ref, typeof, generics,
- Collections – array list, set, dictionary, stack queue
- Serialization – file I/O, regular expressions, JSON, XML
- Exception handling- try, catch, finally, custom exceptions
- Testing- unit testing, xUnit, Fact, theory, TDD
- Debugging – breakpoint, step, logging log level

- SOLID
- Delegates – Func, Action, event, lambda, LINQ
- Multithreading – task, await, async, Thread
- GIT – add, commit, log, pull, push, status, clone

A solution is a container for some related projects

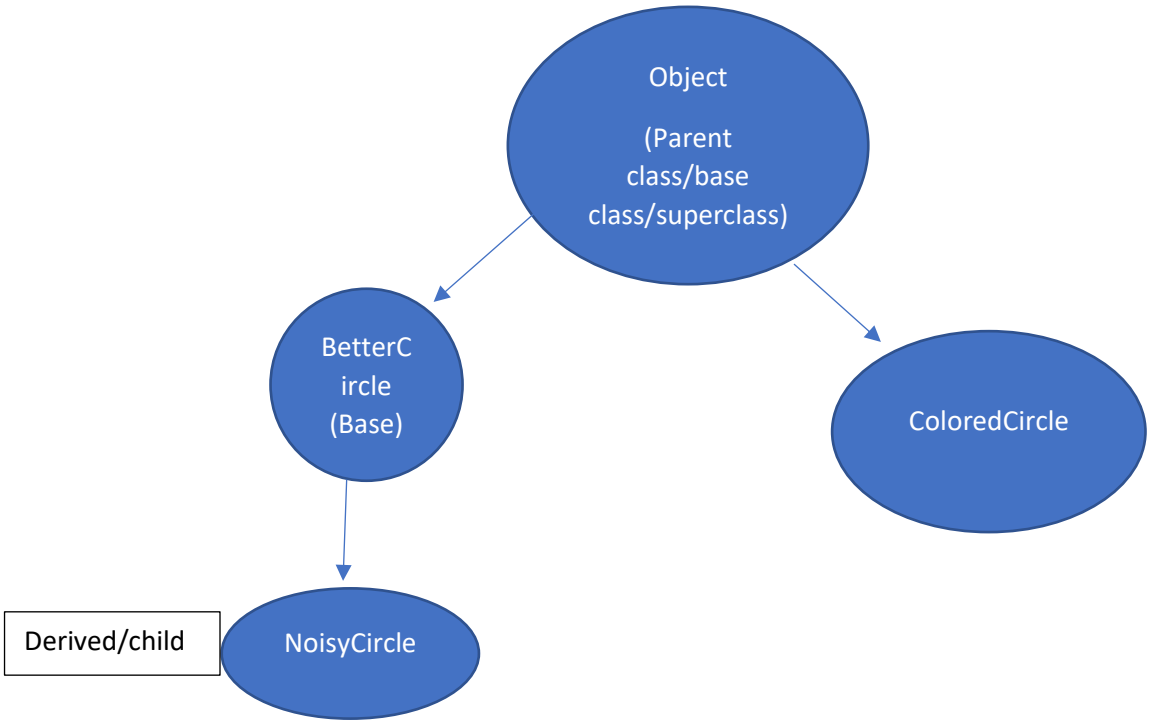
- Project is a unit of compilation and deployment
    - Namespace is a running container for classes
        - Class
            - Properties
                - Methods
                - Variables

Two projects

Console app
(application-has a Main method-starting point of run)
Class Library
(Library-has no Main method – cannot run except in that same application that uses it)

Class: a template to create objects from
Object: a bundle of data and behavior that go well together

Many objects can be instantiated from a single class



Access modifiers

- Private- only this class can access
- Protected-only this class and ANY derived classes can access(even from other projects)
- Internal-only classes in the same project can access(assembly)
- Public- anyone can access
- Protected internal-The type or member can be accessed by any code in the same assembly, or by any derived class in another assembly.

Link with diagrams:
https://stackoverflow.com/a/49597029

Class types (interface, abstract, structs,etc) default for classes is internal
Members (properties, methods, fields, etc.) default for members is private

Wednesday 4/24/19

Git for Windows

Git Bash

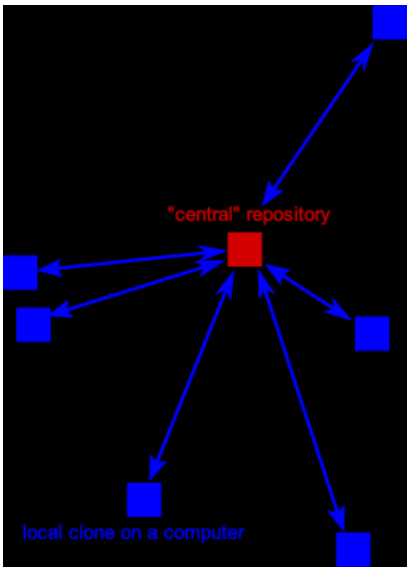
Command line environment

Ls, cd, mkdir

Git is a version control system

Git clone

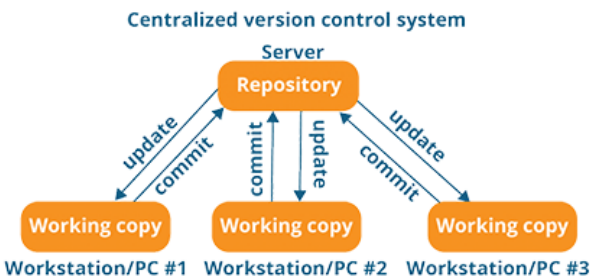
Git pull

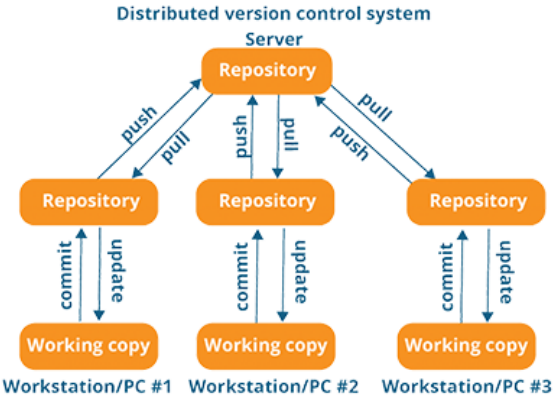


VCS-version control system
2 strategies for VCS

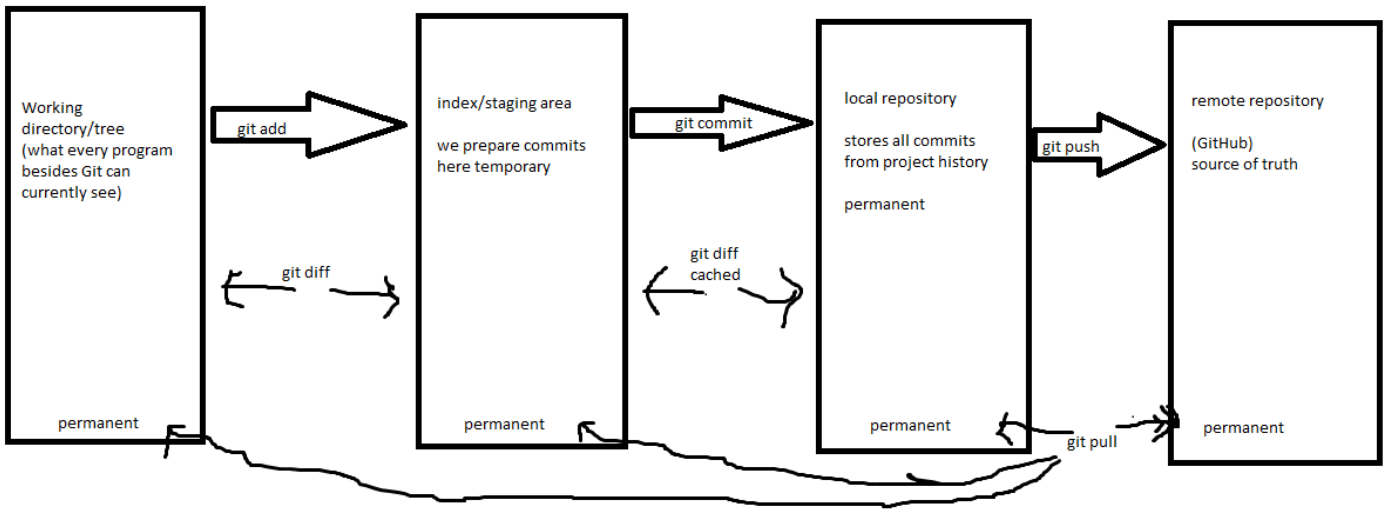
- Central repository – 1 main “central repository”, everyone pushes to it (revature workflow will be similar to this)
  - Subversion (svn)
  - Mercurial (Hg)
  - TFS

- Distributed – everyone has their own copy





trainer-code
.git
Readme.md
notes.txt



. git init

git clone

git log

git push

git add <path>

this will stage all local changes

to any files/folders at/under that path.

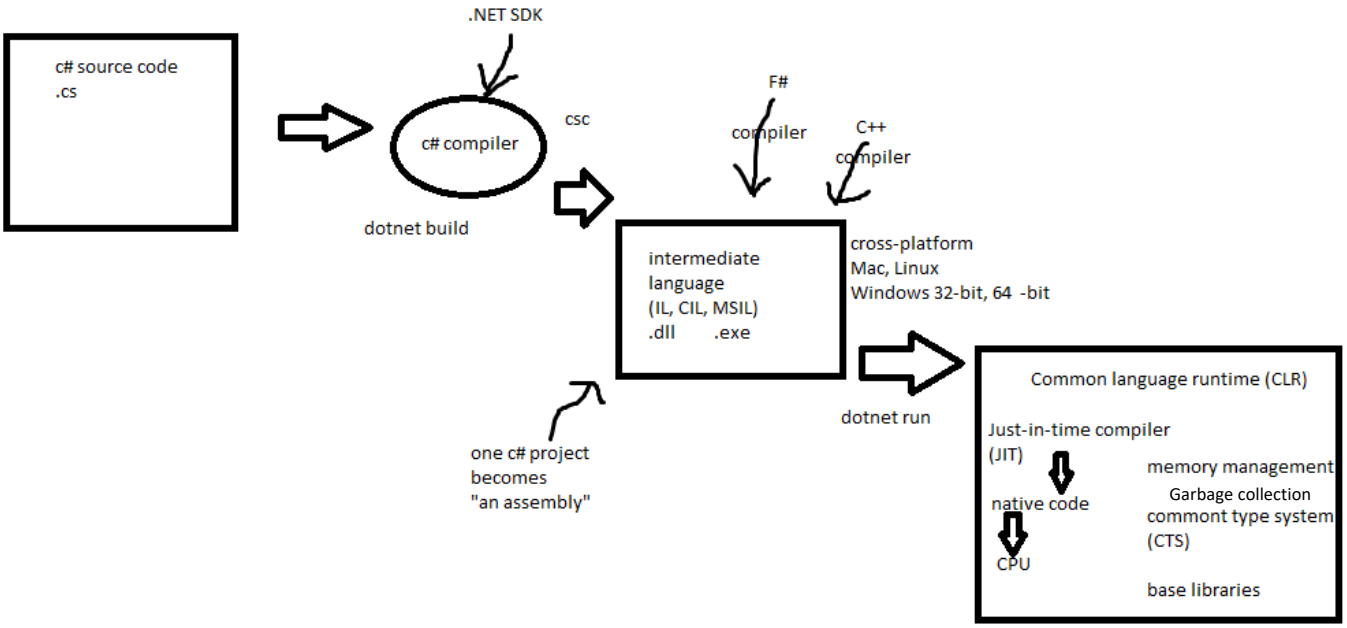
usually: "git add ." from the top-level repo folder

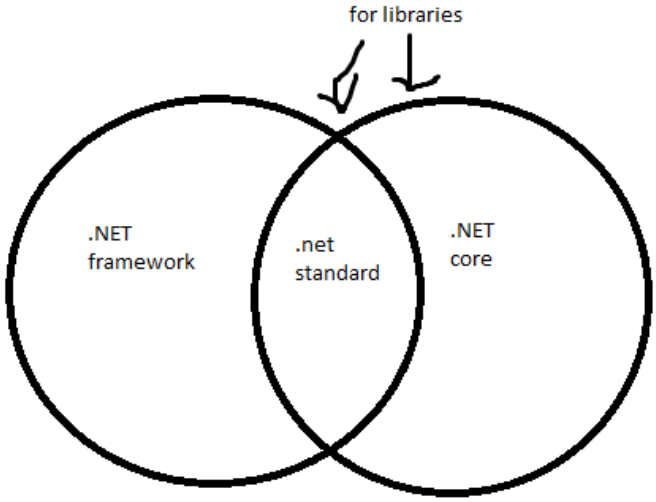
(because "." in bash means, the current directory)

Common Language Infrastructure

- .NET
  - Many languages: C#, F#, VB.NET, C++, Python, Java
  - Common runtime environment
  - Interoperability across languages
  - .NET Framework (4.7)
    - Windows only
    - Has more stuff
  - .NET core (2.2)
    - Truly cross-platform
    - Increasingly as much stuff/libraries as Framework
  - Originally 3rd party port of framework to Mac/Linux
- Any .NET implementation must have

- A virtual Execution System (VES)
  - Framework:CLR
  - Core: Core CLR
- A JIT compiler
  - Core: RyuJIT?
- At least one language compiler
  - Framework CSC
  - Core: Roslyn
- Base class library (BCL)
  - Having collections, exceptions, thread classes, Math, etx
  - Core:CoreFX





Dylan McBee-Quality control

projectsupport@revature

Readiness assessment, weekly evaluation, weekly quality check, panel interviews, resume preparation, project showcase, certification, final check, ongoing project support

Collections

- Arrays
- Arraylist (deprecated)

Generics

- List
- Hashset (doesn't allow duplicates)
- Dictionary a.k.a map (stores values for each key)

Structs
Value type, reference

- Value types stored on stack
- Value types-memory freed up from stack once
- Reference types stored on heap
- Reference types freed up once it is no longer referenced (garbace collection-only for ref types)
- Boxing, unboxing
- Params

Interface	Both	Abstract class
Allows multiple inheritance	Provide general contract to subclasses without 100% implementation	Only one parent class
Nothing can have implementation	Cannot be themselves instantied/constructed	Some(non-=abstract members) can provide implementation to subclasses

	Applies to Class	Applies to Members(methods, variables)
Static	Yes	Yes
Abstract	Yes	Yes
Override	No	Yes
Virtual	No	Yes
Partial	Yes	No
Const	No	Yes
Readonly	No	yes
Sealed	Yes	yes
New	No	yes

Thursday 4/25

Delegates, lambda (look at movie project in VS 2019)

Gitignore
Git reset

4 pillars of OOP

1. Abstraction
    - Separation between necessary functionality and implementation details
        i. C# ex: interfaces, abstract classes, methods, string, double
2. Encapsulation
    - The ability to treat related things as one unit
    - In classes, we can hide members with access modifiers
    - We encapsulate data and behavior into objects
        i. C#:objects, access modifiers
3. Polymorphism
    - The ability to use many behaviors/implementations interchangeably
    - Method overriding: replace parent class' implementation and existing code can use that without any change
    - Method overloading: several methods with same name distinguished by parameters
    - The ability to treat a derived
4. Inheritance
    - The ability to gain structure/data and behavior from another class without duplicating code

OOP sometimes prefer composition over inheritance

Inheritance has an "is a" relationship
Composition is a "has a "

SOLID principles of OOP

- Single responsibility principle
    - Class/method should do only one thing
- Open-closed principle
    - Code should be open for extension, closed for modification
- Liskov substitution principle
    - All objects should be replaceable with instances of their subtypes without breaking the correctness of the program (exception string is a sealed class, don't want people breaking it)
- Interface segregation principle
    - Prefer many small interfaces to few large interfaces
- Dependency inversion principle
    - Depend on abstractions, not concrete classes

Unit testing libraries for C#

- xUnit 3rd party
- NUnit 3rd party
- MSTest Microsoft
  Right click project>new project >search for xUnit>xUnit
  (if getting errors, build project, might still be fetching xUnit)

In xUnit, each test is one method, marked with attribute, either [Fact], or [Theory]
Theory is for parameterized method that can test many values/parameters
Theory has inline data etc.
Fact is for one thing?

Different testing strategies

- Unit testing-test smallest, most divisible part of code, make sure it does what its supposed to (our foundation)
- Integration- make sure whole program works together
- Functional

Like a pyramid: many unit tests on bottom, some integration testing on top of unit tests

3 steps to a good unit test

- Arrange
- Act (the behavior you are testing)
- 3. Assert(verify the behavior was correct)

Reference CollectionTesting VS 2019 solution ^

LINQ, also see collection testing

- Linq stands for language integrated query
- linq has deffered execution for all linq methods that return ienumerable
- linq prefers to return new set of values rather than modifying old one

Linq methods

- Any(true if ANY item in collection is true for the condition
- First
- Max
- Average
- Where
- Select (one of most important, maps values to new values)

Two syntaxes for linq

- Query syntax

```
Var firstLetters2=from x in List
                  where x!=null
                  select x[];
```

- Method syntax

Friday, 4/26/19

Exception handling

- Exceptions are thrown by code that encounters an error and caught by code that can correct the error

Try/catch
Throw -refers to initial/line of code where the error happens(passing the buck), person who gave you the error, not the root cause of the error
Throw exception-refer to exactly this line
Finally

Extension methods

David Fay -HR orientation

HR takes care of onboarding, payroll -timesheets, expense reimbursements, housing deduction, benefits
Promotion ceremony and project deyployment
Employee policy guidance
Ongoing employee relations assistance

Hr email - hr@revature.com
Housing team email – UTAHousing@Revature.com

Biweekly pay every other Friday

Paid 2 weeks arrears
Hired April 22nd
First pay date is may 10th for all time from April 22 through 26
Second pay date may 24 for all time worked from April 29 to may 10

First check is live check

Time entry with revature time and expense portal
Record use of paid time off with timesheet
Deadline for timesheet is Friday 6pm Eastern

Hours worked is hours spent in classroom training and hours outside class in specifically designated learning opportunities

PTO use requires Trainer approval
PTO accrues after 6 months of employment
Eligible for 18 days of PTO earned at a rate of 5.54 hours/pay period

250 relocation expenses if you travel more than 50 miles
$0 if within 50miles

Address used on I9 will be used to calc miles from training location
Certifications will be eligible for reimbursements with trainer approval

Other trainer related expenses may be reimbursed with trainer approval
Earthuware?
Reimbursement processed Tuesdays and Fridays

Benefits effective first of the month after 60 day waiting periods

Life insurance and AD&D
50k coverage for each, company paid-name benefactor

Long term disability -90 day waiting period

401k enroll through adp after first payroll-no company matching atm

Commuter benefit-contribute tax-free deductions for mass transit or parking expenses

Employee handbook
Portal login
Send employee handbook to HR@revature.com

Michael Minton (Recruiter)
2 year commitment starts first day of client project
85% employees after first year get bought out full time by clients

Multithreading

- Only going faster if everything else is still moving, one slows down,everything else does too
- Synchronous -one has to happen after the after (if problem, deadlock happens;multithreading is usually synchronous)
- Threading one at a time across multiple threads at the same time?
- Asynchronous-not waiting for threads to clear up if one slows
- Threading-ability to split into many “lanes” or threads like on a highway
- Downside of multithreading-easy to get a deadlock

Parallel programming-asynchronous

- Deadlocks almost nonexistent
- Downside-Response can be received out of order, write code to handle it(wait for order) ,deal with lost pockets, make sure threads
- Tasks-execution code broken into “packets” that can move between lanes(like cars on highway)
- Async/await-

Look up Task parallel library (TPL library)