Aug 21st

Java

Day one testing how taking digital notes feels.

Might transfer to notebook after

harder to write cleanly without the resistance from the paper.

Will take some practice

This should allow me to mark up slides.

Writing notes on the iPad is stressful because it doesn't look like how I want it to. Might need to lift the pen higher when I write.

Need to turn off double tap to erase, I rarely erase any ways.

Git Introduction

What is it for?

The "Hello World" of Git

First couple of weeks will be Git

↳ Write some review java programs

New topics in two weeks or so.

Moodle → Join teams

Git

Last year we used git for file submission
We didn't really use any of the useful features.

git

Version control system
↳ used to track changes in files
↳ helps greatly in team coordination / collaboration

Program used for tracking your work

gitlab / github are servers that we can use
↳ lets us download and upload

git will let us work together on the same document without stepping on each other's toes.

Team project
↳ Two statements

each person writes their own code but you cannot work simultaneously

Online version of something

Working in tandem can lead to compile errors.
Constantly need to compile and test.
Test the program every few lines or so

If you don't know what your program should do, gotta figure it out

Model

more complex than google drive

One section

download a copy to my computer, lets me test my code

upload or work to our shared server

uploading frequently to git to share it with your ~~teammates~~

Shared version
- ↳ each connecting person will be able to download their own version.

Code overwriting
- ↳ Two people uploading code that changes what someone else wrote

Why is it important to track files?

1 mistakes happen! roll backs are easy

In a large code base, there may be thousands of people working on the code. Important to know who did what

2) If two people make changes to one file don't want the second person to upload to delete the first person's code

3)

3 different copies

Two people to add two methods

One person adds bar1() then uploads
other adds bar2() then uploads

↦ what does the code look like? What should the result be?

should check who

They should have compared it to the most recent server version
↦ see that the method had been added
↦ would have to manually merge

Source Repository System

Source Repository systems are designed to solve this problem by detecting and then managing the conflict between two files.

git push to upload and then the server checks if there are any conflicts

when pulling
- git will try to create a hybrid file if code had been changed by someone else.

Work flow

1) pulls a copy
2) makes their changes (hopefully many small changes instead of one large)
3) They pull again to download any changes to the code base uploaded to the server.

Slide 13 has scenario image

Pull Scenarios

Pull works (no changes)

1)
2 files in code base

x.txt y.txt

A & B download the files at the same time

A changes x.txt and pushes

B changes y.txt and plans to push before doing so, they repull

In this case B now has both files' changes

git does not know the rules of java, only works on a pure text basis

if A deletes code to make B's code not work anymore
git would still let it happen

public methods should stay the same return type
or else you can mess with someone else's code.

Scenario 2

Suppose X.txt contains

10 20 30 40

A & B both download their individual copies

Alice makes it 10 15 20 30 40 and pushes

B adds 25 between 20-30 making it 10 20 25 30 40

What would we want to show up on the file in the server

Desired: 10 15 20 25 30 40

↳ has all of the numbers.

~~git will infer not since~~

git just figures out what the desired files should be by the location that the text was added

if the methods both have the same name, we run into issues. but git will allow it we end up with a compiler error

Scenario 3

Conflict merging

Alice adds 15 between 10 and 20

10 15 20 30 40

Bob also adds 13 between 10 and 20

10 13 20 30 40

What content should show up on the server?

Desired: 10 13 15 20 30 40

Merge conflict

git says: "Too risky to have me to this myself"

We now have to go into the code and communicate with your teammates

If the coders both add code in the same location (same line#) then git won't know how to prioritize or which code to keep.

Lab

```
isLow() {
        if (wheel.getNumber() >= 1 && wheel.getNumber() <= 10)
            return true;
    }

isHigh() {
        if( wheel.getNumber >=19 && wheel.getNumber <= 36){
            return true
        }
        ret false

boolean isEven(int num) {
        if (num>=0 && num % 2 ==0 ) {
            return true;
        }
        ret false

boolean
    isOdd () {
        return ! isEven();
    }

    isRed (
```

Git Introduction

- Spot the bug review
- Review: what is a source repository
- Git commands
- Merge Conflicts

Parallel arrays: element of one array corresponds w/ element have to figure out what the code does

```
public static boolean search(String[] names, int[] ids,
String name, int id) {
  for (int i = 0; i < names.length; i++) {
    if (names[i] == name && ids[i] == id) {
      return true;
    }
  }
  return false;
}
```

i < the total length so it stops before the length

.equals(name)

Trying to check if a name & associated id are both in the same index in their own array

need to use names[i].equals(name);
== only compares addresses

~~ArrayIndexOutOfBounds~~ not

Sometimes it will make string literals the same string (location in memory)

```
public static boolean search(String[] names, int[] ids, String name, int id){
    if(names.length > ids.length){
        for(int i=0; i<names.length; i++){
            if(names[i].equals(name) && id==ids[i]){
                return true;
            }
        }
    } else if(names.length < ids.length){
        for(int i=0; i<ids.length-1; i++){
            if(names[i].equals(name) && id==ids[i]){
                return true;}
        }
    return false;
```

Nested if statement

More nesting = More complex

Parallel arrays suggestion or trick
make it an object?
→ Anti pattern

Suggested

```
public static boolean search ( Student[] students , student search ) {
    for (int i =0; i< students.length ; i++){
        if ( students[i].getName().equals(search.getName()) &&
            students[i].getId() == search.getId() ) {
                return true;
            }
    }
    return false;
}
```

for each loop → better if your searching (reading) through an array

```
for( student s : students){
    if (s.getName().equals(search.getName())) {
        && s.getId()== search.getId()) {
            return true;
        }
    return false;
}
```

Create a method in your class that compares two Students.

Try to avoid messy statements.

for (Student s : students)

Git

Goals of a source repository

↳ version control
  ↳ lets you go back in time to previous code.

↳ allows simultaneous collaboration

↳ Roll back functionality

↳ you have your own copy, not editing at the exact same file.

Logic is the work, not typing.

Working as a team

Option a) One person provides the skeleton (class, main)
  ↳ uploads to git
  The other person will create the ~~methods~~ → print statements
  ↳ ~~skip~~
  ~~duplicate~~

  ↳ still forces member 2 to wait for the first person

Option b) Two people together work on the class/method/header (structure)
  Then they both add their own bits on their own

Option c) Both write the whole program & merge them together

Adv of team work
  ↳ division of labour

Scenarios

G senerario

Remember that all merge "fun" happens because two people download the code to their computer at aprox. the same time

One person makes a change (locally + first) and then pushes their work to the server

Suppose that a file initially looks like

```
public class methods {
    a() {}
    b() {
    c() {}
    d() {}
}
```

//=comment

A// → a() {}

b() {

B// →

C// → c() {}

d() {}

}

Alice adds a comment before b() and method d() and pushes

Bob adds a comment before c()

what happens when Bob commits.

Git will not delete people's work

Ideal case! (git perspective)

Server version

```
// comment
    a() {}
    b() {}
// comment
    c() {}
    d() {}
```

Bob's version

```
a() {}
// comment B
b() {}
c() {}
d() {}
```

Bob's version after pull

```
// comment A1
a() {}
// comment B
b() {}
// comment A2
c() {}
d() {}
```

So long as code isn't being added to the same place, the merge SHOULD work

Unit testing

Scen 2

Bob & Alice both download Methods.java

b[] {X is wrong value}:

Alice makes x=20 and pushes

Bob makes x=10 and pulls

What happens when Bob pulls?

This is a merge conflict & is prompted to make changes to the program

Before you start coding, you should pull!

Java Review

Readability
Git Pull / Git Merge
Demo
Review

→ Thursday Lab gives a merge conflict.
→ Branching in a few weeks

As you add more features to a program, it is likely

- to become disorganized.
  - the bonus would miss the initial planning phase of coding.
- the fastest way to add commands you might be adding to the main or other method ends up messy

* Lots of short methods is best.

Too long method
- if it's doing more than one thing.

Method for getting Ready() would be one method because all your morning steps are for one goal.

Rule of Thumb:

→ Fits on one screen length.

Scrolling up and down to figure out a method sucks.

Good length for a method → 5 lines

About half a screen length per method

This never checks that you have money

```
int money = 100;
while (shouldPlay) {
  System.out.println("Enter a bet or 0 to exit");
  int amount = reader.nextInt();
  if (amount > 0) {
    System.out.println("What number would you like to bet on?");
    int guess = reader.nextInt();
    wheel.spin();
    if (guess == wheel.getValue()) {
      money += 35 * amount;
    }
    else {
      money = money – amount;
    }
  }
  else {
    shouldPlay = false;
  } }
```

} and check in helper.

} in helper

Exercise:
Think of some useful helper methods!

if pulling out the while if statement, you need to have input and variable initialization

public static ~~boolean~~ int ~~isValidBet()~~ bet

```
int money = 100;
boolean shouldPlay = false;

SOP("Enter your bet or 0 to exit");
int amount = reader.nextInt();
if(amount > 0){
    shouldPlay = true;
}
while(shouldPlay){

  money = playRound(amount, wheel);
}
```

public static int playRound(int amount, int bet){

Dan's suggestion

```
int money = 100;
while (shouldPlay) {
  System.out.println("Enter a bet or 0 to exit");
  int amount = reader.nextInt();
  if (amount > 0) {
    money = playRound(money, amount, wheel);
  }
  else {
    shouldPlay = false;
  } }
```

descriptive method names will help

Check the slides for the continuation of

use a constant w/ a descriptive name to avoid magic numbers.

final int ROULETTE_ODDS = 35;

your first draft will be messy, but you need to work on refactoring.

ternary operator (? :)

Git Sequence Review

git add	A	← can do multiple
git commit	C	← can do multiple
git pull ←	P	← as frequently as possible
git push	P	

All computer programmers Program

you can revert to a commit (kind of a restore point)

More frequently you pull, the easier it is to manually merge.

Git Scenarios

Alice & Bob both download the same file

Alice pushes her work monday

Bob pushes his stale file

3) Alice adds foo() before the main method Push

↳ Bob does not pull and adds foo() after main and pushes

What happens? Successful Merge

They are both added to their locations respective to the main method.

This is fine because they changed things in different locations, but the compiler is not happy.

After pulling recompile and check

⇒ eventually we will be able to test when pulling.

4) Alice deletes a folder called fun from the source ~~push~~ pulling

Bob tries to add a file to the folder

Merge conflict

Their work is incompatible.

Demo

Make a folder to clone into for both folders.

user 1) Alice adds a testfile.txt w/ text from one computer and ten acpp's

user 2) Pulls → change is added.

Style

user 1) changes line 3 and pushes her code. ACPP

user 2) Changes line 3 ACPP → gets to pull

Merge Conflict.

<<< HEAD
=====
merge conflict
>>> <alphanumeric shit>

Ungraded Lab

if you don't want to import a package you can fully qualify it

Random rand = new Random(); ← Doesn't work w/o import

java.util.Random rand = new java.util.Random();

↳ This works but is ugly.

Tells the program where to go to find the package when you call it.

You normally import

BUT you fully qualify if you had two classes w/ the same name BOTH imported into the program

Java has specific rules about packages

if the package name is "secondpackage" the folder must have that name.

if you have a dot in the package name it will represent a sub folder

animals.mammals

↳ This is the folder mammals in the animals package

Java Review

Sep 1st 2022

Package must be in a folder w/ the packagename.

Topic List

- Objects
  - Errors
  - Dynamic Arrays
  - Method overloading
  - Enums
  - Multi Dimensional Arrays
  - toString()
  - CLI compiling / running

Exception : throw new <excName>();

try / Catch

Enums
- ↳ Constants w/ meaningful name

Objects
- ↳ Constructor
- ↳ (Private) Fields
  - ↳ Getter / Setters
  - ↳ instance methods
  - ↳ Encapsulate our logic
  - ↳ Arrays, dynamic, 2d array
  - ↳ Algorithms : min of an array, Searching, Sorting
  - ↳ Loops

Why might we want to define our own object?

→ To represent a real world object in a program

→ To store associated data of varying types together.

→ To be able to reuse our object in multiple programs w/o having to rewrite it.

→ Clarity / Readability

Add methods that encapsulate or hide how the

Ex over

```
public class Dog {
    public int ageInPeopleYears;
    public int ageInDogYears;
    public String name;
    public String breed;
}
```

Should all be private

```
Dog d = new Dog();
d.ageInPeopleYears = 10;
d.ageInDogYears = 70;
d.breed = "Labrador";
d.name = "Spike";
```

Dog class has little control

Not always getters & setters needed.

example on slide 10;

```
public int getHumanYears() {
    return this.ageInHumanYears;
```

if two fields are related connected by a formula, store one value then calculate it in your getM

Q1 It is an instance method because it is not static

Q2 An instance method MUST be called on an object

Static `Math.pow(3,2);`

Class name

Inst `d.getName();`

object name

Import

↳ does not affect this rule

↳ Tells the computer where to find the class

How can people take this long on get & set methods?

More pen testing

I like the rounding

```
public class Rectangle {
    private double length
    private double length;

    public double getLength(){
        return this.length;
    }

    public void setLength(double length)
        this.length = length;
    }

    public double getWidth(){
        return this.width;
    }

    public void setWidth(double width ){
        this.width = width;
    }

    Public Rectangle(int length, int width){
        this.length = length;
        this.width = width;
    }
}
```

getLength

length

length

length

Ball pen

2.5 fountain

Personal excercise

toString(){

```
Rectangle r = new Rectangle(4,5);
Rectangle r= new Rectangle(4,5);
// Rectangle r = new Rectangle();
Rectangle r = new Rectangle();
r.setWidth(4);
r.setLength(5);
r.getWidth();
r.getLength;
```

Every pull should be retested git might not notice.

Junit & Unit testing

Sept 9th 2022

Unit testing Theory

JUnit basics & Maven

```
for(int[] a: ar){
    {for(int i : a){
        System.out.println("i");
    }
```

Test Details

Snippets
Code reading

```
int[][][][][] x = new int[4][2][3][2][5];
```

int[][][] y = x[1][0]

int triple array.

3d int array.

S.O.P(x[0][0][2].length); → 2

Shows code and asks what it does / fix it

Piecewise coding.

No unguided coding.

Sean

Sam goes to pull, they realize there is updated work.

Automerge succeeds but Sam needs to retest everything

Already up to date, no changes.

Make testing as easy as possible

Two problems

- Too slow to test, what if Sam finishes and another change is made.
- Very difficult to test other people's code since you might not know the details of the components of the program

More generally

- Testing has to be done constantly. We end up testing the same code all the time
- Can't test only at the end, you'd be fruuucked.
- Even if you do have no bugs, you will still need to update and maintain it over

Refactoring Sequence

- git commit
- modify a bit of code
- test the code → unit testing is meant to make this easier
- if it works, git commit & repeat
- if it doesn't work consider rollback, comparison, etc.

Any time you make changes to a module, you need to ensure two things:

- The change has been added.
- Test the OTHER public behaviours.

Unit Tests

Write a bunch of methods to test our classes

These methods will call all of the public methods within our class.

Test ex slide 12

```
public static void main (String[] args) {
    Rectangle r = new Rectangle(1, 2);
S.O.P( r.getHeight() );
S.O.P( r.getWidth() );
S.O.P(r.getPerim());
S.O.P((r.getWidth + r.getHeight())**2);
```

<- should print 6 <- Check same.

* if we know what the height should be then if it gives another answer we know something is wrong.

that these results are the

Then do some with area.

Test Slide 12 cont.

Generally avoid 0, 1, 2 since they can give similar answers for different operations.

Not using the same values in the constructor.

Dan's suggestion

```
main(){
    Rectangle test = new Rectangle(2,3);
    S.o.p(test.getArea()); // should print 6
    S.o.p(test.getPerimeter()); // should print 10
    S.o.p(test.getLength()); // should print 2
    S.o.p(test.getHeight()); // should print 3
```

it checks

Later

Skeleton Methods. Throw exceptions instead of returning random place holder.

Have to hardcoding so that we can see if the output matches our expectations.

Void methods are harder to test. ex. mutators.

This example is easy because there are no branching methods.

JUnit we would have multiple tests

Be careful when choosing

```
assertEquals(2, test.getWidth());
assertEquals(<value>, statement);
```

checks if <value> == <statement return>;

if true continues, else throws an error

l^2	$l+h$	[redacted]
$l+2$	$h+2$	$(l \times l + 2h)$
$2l$	$2h$	

Lab Review

JUnit aids us in automating tests.

The library includes
- utility methods
- driver main method

Write one method per test
- finishes without exception => passing
- if it generates an exception or times out => failing.

Xml → markup language

Roll backs don't move back in time, but it is changed and logged in the history moving forward.

Unit Testing & JUnit

Sept 10th 2022

Object Class.

Next week: Inheritance.

Readability Example

3 improvements.

```
public class Square {

    private double width;
    private double length;
    private double area;

public Square(double l, double w) {
    if( l != w) { throw new IllegalArgumentException("Not a square")}

    this.length = l;
    this.width = w;
    this.area = l * w;

// getters

    public double getArea(){
        return this.area;
    }

    public double getPerimeter(){
        return l + w + l + w;
    }
}
```

don't store things if they can just be calculated.

give them REAL names.

only take ONE — no need for both.

w/ only one you can do it: if (l ≥ 0) {

why have both length & width

① do the arithmetic in the function ①

inputs are irrelevant

this.length * 4;

If you had a set

Unit test before prod code

Relationship to Unit Testing

We will need to refactor our code

Automated testing is known as a "test harness"

- Run your tests, (all should pass)

- Refactor the code a small amount
- Rerun the tests and see if they pass
- Repeat until refactoring is done.

Adv of making a unit test.

↳ JUnit would be standardized.
↳ lets you use asserts.

Most languages have something like JUnit.

Unit testing => JUnit testing.

Purposely make a bug in Prod and see if my tests still pass
↳ Test the tests.

Testing Guidelines

- Each method should have it's own set of tests (each branch needs a test)
- Some methods should have several asserts but should all be related to the same scenario.
- Each test should be independent Do not assume anything about the order that tests run.

Remember that tests are also coded. All readability rules still matter.
↳ Magic numbers are okay in most cases.
↳ This is so that you know and can predict what the outcome is. describing numbers is a good idea

Multiple readability rules that conflict sometimes

@Test (assertEquals() ← Library package methods

assertEquals()
↳ usually something before but it is statically imported so that you can import a method.
Not Recommended

assertTrue(x == y) = assertEquals(x, y)
↳ be careful of ordering for readability and debugging.
↳ considered more readable

assertEquals(x, y, "Failure Message");
↳ overloaded.
assertEquals(x, y, 0.0000001); ← constant diff has to be less than

doubles issues

x = 1.99999
y = 2
← if this came from a computation it should often be 2, but rounding fucks it.

1/3 = 0.33333

Don't check if doubles are equal, can differ by rounding.

if ((double - double) < 0.0001) { These are the same }

if (Math.abs(x - y) < 0.000001) ← Treat as equal.
Most cases this is fine but it might be different due to required precision.

Dependent tests ← BAD slide 10 example

```
public class AppTest {

    final double TOLERANCE = 0.0000001;

    private static Rectangle r = new Rectangle(2,3);

    @Test
    public void testArea(){
        assertEquals(6, r.getArea(), TOLERANCE);
        r.setLength(10); // preparing the next test.
    }

    @Test
    public void testPerim(){
        assertEquals(26, r.getPerimeter(), TOLERANCE);
    }

}
```

put all of your variables IN the tests

if all run together
← passes
← assigns
← passes

if run alone
← fails

In almost every case, tests should not be dependent.

Blackbox Vs Whitebox Testing.

Whitebox looks at the prod code and figures out the best tests

Blackbox figures out the results they want and THEN build your code to solve.

Whitebox
— Seems more approachable for less experienced
↳ easier to know exactly which tests you need to write
↳ lets you see branching.
↳ Can lead to something in test & prod.

Blackbox
↳ Seems → simplifies writing code
↳ if goal is simple but the way to get there is complex.

Testing Exceptions

```
public Rectangle (double length, double width) {
    if ( length <= 0 || width <= 0) {
        throw new IllegalArgumentException("Invalid");
    }
}
```

Generally want all our tests to pass.

We can catch it.

Try/Catch → if we reach the end of the try then the production code failed (since it didn't throw)

→ read the slide ~~#~~ 16/17

fail("Failure message") ← makes your test fail.

Monday inheritance

More Unit testing.

Checks if a string is a number

↳ first part checks if it is negative ↙ my fix

```
if (s.charAt(0)=='0' || Character.isDigit(s.charAt(0))){
    for (int i=1; i<s.length(); i++){
        if (Character.isDigit(s.charAt(i))) {
        }
        else isValid=false;
    }
```

can make at start at 1 or 0.

move into if and negate

digit was invisible

```
    return isValid;
}
```

In orgiinal Problem

Two for loops :

empty if followed by an else, should flip and negate the if.

!(a||b) = !a && !b

demorgan's law

Object Class

Why does println know to use the .toString?

- Every object automatically has a toString that returns a String
- The method println

objects have other automatically defined

.equals

public boolean equals

Slides going too fast today

@Test

```
Square s = new Square(2);
Square bigger = s.doubleMe();
Square max = new Square(4);
assertEquals(max, bigger);
```

The test will fail

Inheritance

All classes automatically inherit methods from a library class Object

.toString → defaults to return the memory location

.equals → compares the reference value

Several More

We can override these methods in our own class

overloaded → same method name, different inputs

overriding → exact same header.

Motivations

Class Book

3 fields Author, title, date

There are some methods.

Now, suppose you want to define a type ElectronicBook

Electronic books wants to store all info that books store, but it also wants to store more.

We can use inheritance.

```
public class ElectronicBook extends Book {
    // all the fields & methods from book are inherited.

}
```

if we add stuff to the book class, Electronic books is also updated.

```
Public class ElectronicBook extends Booth
    private int fileSize;                    ← has four fields because it is inheriting

}
```

privat fields are not inherrited.

Basics of inheritance

Test will be not this monday but the next monday

- protected
- constructors
- polymorphism

Don't need to store .class file in gitlab

↳ .class will almost always create a merge conflict

omit those files from the source repo

.gitignore

< list all the kinds of files you don't want to include >

*.class

always git status if you are confused or lost.

By default, it extends the Object class

```
public class EBook extends Book {

    private int size;

    public int getSize() {
        return this.size;
    }
}
```

← has all of the book class fields/methods

If a field is private, it cannot be directly accessed by any other class

This includes in derived classes

classes that inherit from

Access from the subclass w/ getters

Can use protected instead of private in book.

protected means visible to the class itself, any derived class, plus any classes part of the same package.

if we put everything in on package, protected essentially becomes public.

Not always correct to make a field visible to a derived class.

Constructors

Needs all parameters for a constructor.

EBooks has all of these fields but one not initialized.

Needs to create all the fields, but EBooks don't know how to call them.

Super()

```
public class EBook extends Book{

    public EBook(String author, String title, int year, int size){
        super(author, title, year);      <- has to be the first line in the constructor
        this.size = size;
```

```
public class Tiger extends Cat{

    private int numStripes;
    public Tiger(int age, String color, int numStripes){
        super(age, color);
        this.numStripes = numStripes;
    }

}

public class BlackPanther extends Cat{
    private double speed;
    public BlackPanther(int age, double speed){
        super(age, "Black");
        this.speed = speed;
    }
}
```

Polymorphism

When we define inheritance relationships, we create what is known as an is-a relation ship.

a derived type can be stored in a parent type.

```
Cat c = new BlackPanther(5, 1000);

Base type = new derivedType();

BlackPanther bp = new Cat(3, "orange");   ← fails
    ↳ need precise black panther

Cat c = new BlackPanther(3, 100);
c.getSpeed();   ← Not allowed.
    ↳ since getSpeed isn't in the parent, it doesn't work
        Compiler error.
```

Can override parent methods in the derived class.