

Testing Enigma

Lab 6

Announcements

- ▶ Lab 6: Testing Enigma
 - Due Friday, 02/25
- ▶ Project 1: Enigma
 - Checkpoint: Friday, 02/25
 - Due: Friday, 03/04
- ▶ Homework 4
 - Due: Tuesday, 03/01

Gitbugs!!!

You need to submit one for this assignment.

This is worth 0.5 points of your lab grade.

Follow the directions in the [spec](#).

Alphabet

- ▷ Represents a set of possible characters
- ▷ Is indexable

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

H	I	L	F	N	G	R
0	1	2	3	4	5	6

Permutation

- ▷ Represents a map of characters in a cycle
- ▷ Only contains characters in the alphabet

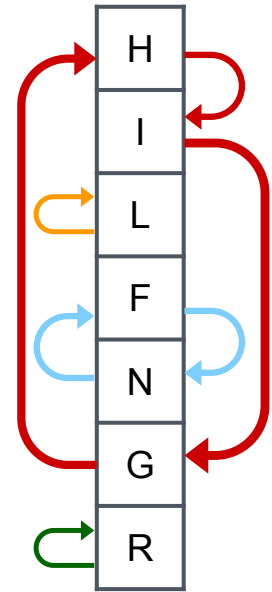
Permutation

- ▷ Represents a map of characters in a cycle
- ▷ Only contains characters in the alphabet

Alphabet

H	I	L	F	N	G	R
0	1	2	3	4	5	6

Permutation (HIG)(NF) (L)



Permutation

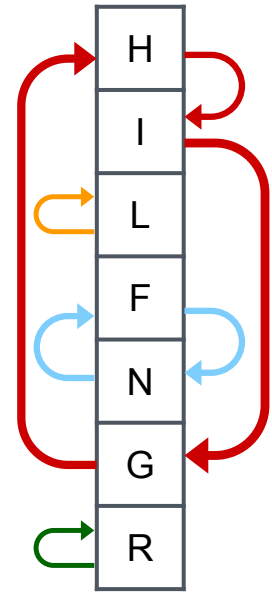
- ▷ Represents a map of characters in a cycle
- ▷ Only contains characters in the alphabet

Alphabet

H	I	L	F	N	G	R
0	1	2	3	4	5	6

Permutation (HIG)(NF) (L)

chars in Alphabet
but NOT in a cycle
maps to itself



Permutation

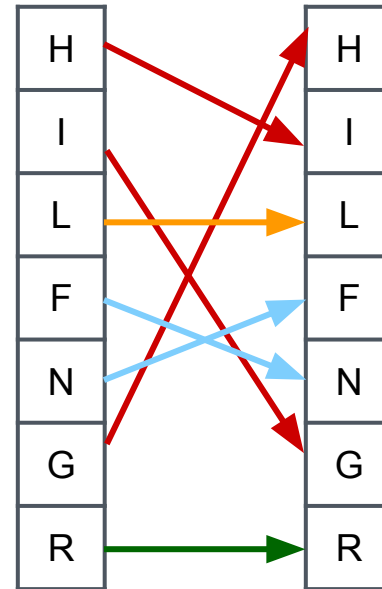
Alphabet

H	I	L	F	N	G	R
0	1	2	3	4	5	6

Permutation

(HIG)(NF) (L)

permute



Permutation

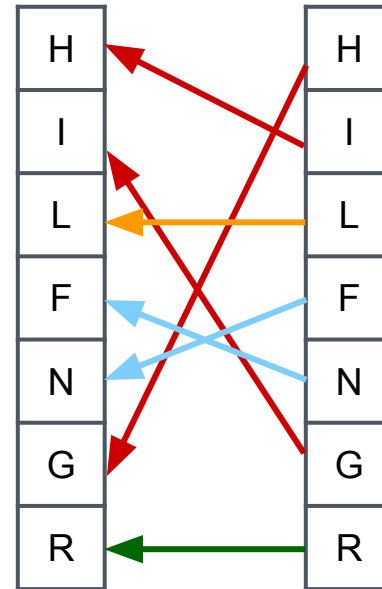
Alphabet

H	I	L	F	N	G	R
0	1	2	3	4	5	6

Permutation

(HIG)(NF) (L)

invert



What is testing?

What is testing?

- Program correctness - does the program do what you think it's supposed to do?

What is testing?

- Program correctness - does the program do what you think it's supposed to do?

“Untested code is
broken code.”

Why is testing?

Writing tests takes development time. Why is it worth spending this time, when it could instead be used to develop new features?


Why is testing?

Writing tests takes development time. Why is it worth spending this time, when it could instead be used to develop new features?

- Confidence that code is correct
- Guarantee that adding features won't break existing ones
- Bugs are found before they are released
- Code that is easy to test is often well-designed
- Insights into performance
- Easier to diagnose bug causes

How is testing?

- Many different *kinds* of tests
- Correctness tests
 - Unit tests (test *individual behaviors*)
 - Integration tests (test *interactions between components*)
 - End-to-end tests (test *entire systems*)
 - etc.
- Performance tests
 - Many sub-categories as well, but won't discuss now

 We focus on these two

Unit Testing

- ▷ Tests one “unit”
- ▷ Testing per-method
 - One method, many cases
- ▷ Testing per-case
 - One case, (possibly) many methods

Unit Testing

```
/* ***** TESTS ***** */  
  
@Test  
public void checkIdTransform() {  
    Alphabet alpha = getNewAlphabet();  
    Permutation perm = getNewPermutation("", alpha);  
    checkPerm("identity", UPPER_STRING, UPPER_STRING, perm, alpha);  
}
```

Unit Testing

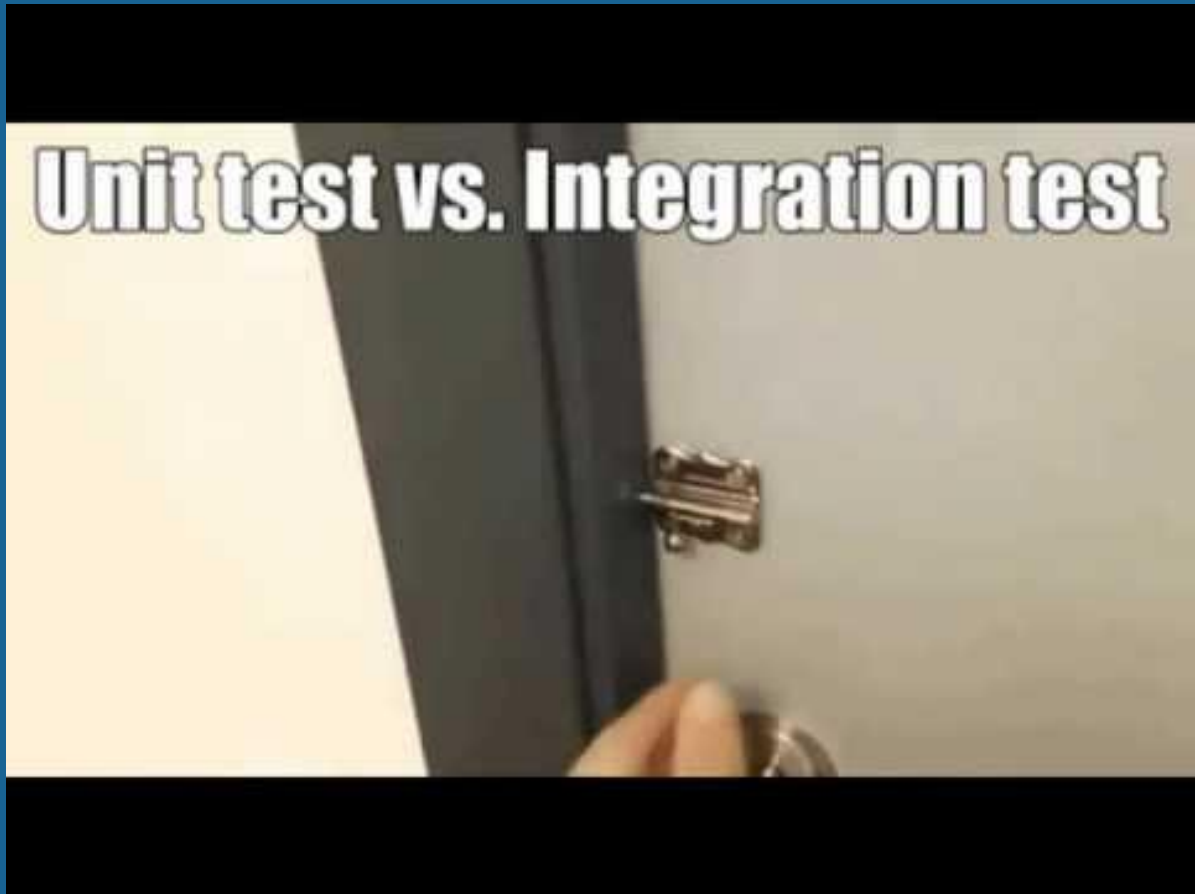
```
/* ***** TESTS ***** */
```

```
@Test  
public void checkIdTransform() {  
    Alphabet alpha = getNewAlphabet();  
    Permutation perm = getNewPermutation("", alpha);  
    checkPerm("identity", UPPER_STRING, UPPER_STRING, perm, alpha);  
}
```

Example of
per-case testing

Acceptance Testing

- ▶ Why do we need it?



Source: <https://www.youtube.com/watch?v=0GypdsJulKE>

Acceptance Testing

- ▷ Why do we need it?
- ▷ Each unit worked perfectly separately
 - *Completely* failed together!
- ▷ Much like your code — many moving parts
 - Object interactions
 - Complex code fitting together

Acceptance Testing

.conf

+

.in

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
5 3
I MQ      (AELTPHQXRU) (BKNW) (CMOY) (DFG) (IV) (JZ) (S)
II ME     (FIXVYOMW) (CDKLHUP) (ESZ) (BJ) (GR) (NT) (A) (Q)
III MV    (ABDHPEJT) (CFLVMZOYQIRWUKXSG) (N)
IV MJ     (AEPLIYWCOXMRFBSTGJQNH) (DV) (KU)
V MZ      (AVOLDRWFIUQ)(BZKSMNHYC) (EGTJXP)
VI MZM    (AJQDVLEOZWIYTS) (CGMNHFUX) (BPRK)
VII MZM   (ANOUFFRIMBZTLWKSVEGCJYDHXQ)
VIII MZM  (AFLSETWUNDHOZVICQ) (BKJ) (GXY) (MPR)
Beta N    (ALBEVFCYODJWUGNMQTZSKPR) (HIX)
Gamma N   (AFNIRLBSQWVXGUZDKMTPCOYJHE)
B R       (AE) (BN) (CK) (DQ) (FU) (GY) (HW) (IJ) (LO) (MP)
          (RX) (SZ) (TV)
C R       (AR) (BD) (CO) (EJ) (FN) (GT) (HK) (IV) (LM) (PW)
          (QZ) (SX) (UY)
```

```
* B Beta I II III AAAA
HELLO WORLD
* B Beta I II III AAAA
ILBDA AMTAZ
```

.out

```
ILBDA AMTAZ
HELLO WORLD
```

(or error)

Acceptance Testing

- ▷ Use `staff-enigma` to generate test cases for testing your own code
- ▷ See project 1 spec for how to run acceptance tests in IntelliJ

Which one should I do?

- ▶ **HIGHLY recommended to do BOTH + more!!!**
 - This will save you so much time when doing the actual project
- ▶ “I’m just starting out with the project” or “I want to make sure my implementation is right before moving on”
 - Unit testing
- ▶ “I’m already past Alphabet/Permutation” or “I want to get a better understanding of how the machine works before starting”
 - Acceptance testing

Grading

Write tests to catch bugs in our sample buggy implementations!

- ▶ Unit testing
 - Catch 8/12 buggy implementations
- ▶ Acceptance testing
 - Catch 5/8 buggy implementations
- ▶ If your tests fail on any correct implementation, you'll get 0 credit

Sample Gitbug submission - worth 0.5 points!

Project Collaboration

We encourage you to work with a partner to understand Enigma, the testing framework, and to brainstorm tests and edge cases!

However, if you plan on using these tests in your project, **you must do the code-writing yourself.**