Variable Usage
Making HTTP requests

Advanced Java
Maps, Sets, Lists
Exceptions

Use your VCS
Lawrence Angrave
Finish the sentence

- Initialize each variable ...

A) as early as possible.
B) as it is declared.
C) if necessary.
D) before every use.
Which is better?

A
public void foo(int [] A) {
    for (int i = 0 ; i < A.length ; i ++) {
        ...
    }
    ...
    for (int i = 0 ; i < A.length ; i ++) {
        ...
    }
}

B
public void foo(int [] A) {
    int i;
    for (i = 0 ; i < A.length ; i ++) {
        ...
    }
    ...
    for (i = 0 ; i < A.length ; i ++) {
        ...
    }
}
**Which is better?**

A

```java
int xPos = getPositionX();
int yPos = getPositionY();

int xTranslated = translate(xPos, XMATRIX);
int yTranslated = translate(yPos, YMATRIX);

System.out.println(xPos + "->" + xTranslated);
System.out.println(yPos + "->" + yTranslated);
```

B

```java
int xPos = getPositionX();
int xTranslated = translate(xPos, XMATRIX);
System.out.println(xPos + "->" + xTranslated);

int yPos = getPositionY();
int yTranslated = translate(yPos, YMATRIX);
System.out.println(yPos + "->" + yTranslated);
```
Which is better?

A

String nameString = getName();
System.out.println(nameString + ":" + lookup(nameString));

nameString = getAlias();
markUsed(nameString);
return nameString;

B

String nameString = getName();
System.out.println(nameString + ":" + lookup(nameString));

String aliasString = getAlias();
markUsed(aliasString);
return aliasString;
public class Variables {
    private static String invalid = "INVALID";
    private static String[] args;

    public static String handleIt(String argString) {
        args = argString.split(" ");
        return handleArgs();
    }

    public static String handleArgs() {
        int strLength;
        for (int i = 0; i < args.length; i++) {
            strLength = args[i].length();
            if (strLength > 5) {
                return args[i] + ": " + strLength;
            }
        }
        return invalid;
    }
}
Exceptions

- Events that occur during program execution
- Disrupt the normal flow of the program
  - (e.g. divide by zero, array access out of bound, etc.).
  - `throw new Exception("Fix Your Code!");`
- In Java, an exception is an object that wraps an error event
  - contains information about the error including its type
- Typically handled through the use of `try/catch`
- Important piece of the interface of a method
  - Method signature includes what exceptions it might throw
Java try-catch-finally blocks

```java
try {
    code that may throw exception
    ab c() de()
} catch(Exception ex) {
    ex.getMessage()
}
finally {
}
```
"A" "B" "C" "D" or something else?

```java
func1() {
    blahblah...  -- throws IOException
    System.out.println("A");
}

func2() {
    func1();
    func1();
    System.out.println("B");
}

main(...) {
    try {
        func2();
        System.out.println("C");
    } catch(Exception ex) {
        System.out.println("D");
    }
    // "E" = Any other output.
}```
Downloading from the internet

- **Hyper Text Transport Protocol (HTTP)**
  - Client-Server Model
    - Client makes requests, server responds

- **Uniform Resource Locators (URL) to specify server/resource**
  - `http://host.name.here:port/path/to/resource` (or https)
  - Port defaults to 80

- Takes a “command”: GET is standard for getting data
  - Assumes idempotency, can be cached

- **Returns:**
  - Status code: 200 (OK), 401 (Unauthorized), 404 (Not Found)
  - Response body
In Java

- Easy with the power of libraries
  
  1. java.net.URL

  ```java
  URL url = new URL("http://google.com");  // throws MalformedURLException
  
  InputStream inStream = url.openStream();
  InputStreamReader reader =
      new InputStreamReader(inStream, Charset.forName("UTF-8"));
  ```

*https://docs.oracle.com/javase/7/docs/api/java/net/URL.html*
In Java

- Easy with the power of libraries
  1. java.net.URL
     URL url = new URL("http://google.com");  // throws MalformedURLException
     InputStream inStream = url.openStream();
     InputStreamReader reader =
        new InputStreamReader(inStream, Charset.forName("UTF-8"));

  2. com.google.gson.stream.JsonReader
     JsonReader jsonReader = new JsonReader(reader);
     Gson gson = new Gson();
     Thing thing = gson.fromJson(jsonReader, Thing.class);

*https://docs.oracle.com/javase/7/docs/api/java/net/URL.html
Java Exception Hierarchy
List Set Map semantics

- **List**
  - `add`
  - `get`
  - `size`

- **Set**
  - `put` $O(1)$
  - `contains`
  - `remove`

- **Map**
  - `key -> value`
  - `put(k, v)` $O(1)$
  - `get(k)`
  - `containsKey(k)`
java.util.List & ArrayList

List<String> items = new ArrayList<String>();

items.add("Apples")
items.size();

list.get(0); // throws IndexOutOfBoundsException!
list.get(list.size() - 1);

for(String item : list) { ....}
java.util.Set & HashSet

Set<String> keys = new HashSet<String>();

keys.add("Apples")
keys.contains("Apples")
keys.remove("Apples");
java.util.Map & HashMap

Map <String,String> acronyms
    = new HashMap <String,String> ();

acronyms.put( "LOL", "Laughing out loud")

acronyms.get( "LOL" )
VCS

- import
- export
- checkout
- update + commit cycle
How often should you commit?

A) Every 40 hours work (once a week)
B) Every 8 hours of work (Once every day)
C) Twice a day (every 4 hours of work)
D) Every hour
E) > once per hour
To Dos for Thursday

- Read Ch. 8 (Defensive Programming)
- Figure out what URL pulls pages other than page 1 from themoviedb.org (i.e., read API documentation)
- Extend your project to load a specified number of movies
  - The # of movies is a parameter
  - Make server requests until you have enough movies
  - If the user requests more movies than there are, return as many as there are.
  - More details coming in Piazza

- Cleanly handle exceptions & invalid inputs