Objectives

• Implementing game logic code separate from UI code
• Working with HTTP REST APIs
• More creative freedom with images & audio
Adventure Antipatterns - Static

• Making game variables static is tempting...
  • Runtime programs lend themselves to static
• However, what if you want to run multiple instances at once?
  • static would force all instances to share the same values
• Primitive constants can *always* be static
• Object constants (final object variables) *may* be static, but evaluate the side-effects of doing so
  • E.g. a player inventory list
Adventure Antipatterns – Recursion

• Recursion can be used to replace loops
  • Works very well for problems with subtasks, like traversing a tree

• Adventure has a single-step in -> single-step out design
  • Recursion won’t present a result until the “looping” is done
  • Can’t return anything except for a final result from that function unless you introduce side effects
  • I.e.: String evaluateCommand(String input)
Adventure Antipatterns – Multiple Responsibilities

• S of SOLID – Single Responsibility principle
• Consider a theoretical function

```java
void executeGo(String rawInput)
    validate direction & get room
    change room
    generate feedback text
    print feedback text
```

• Printing is not part of your Adventure game model; it is part of the CLI
Adventure Antipatterns - Testing

• Use your single-step input function
  • Pass in input, check output / public functions

• Edge cases
  • Mixed casing
  • Moving in a bad direction
  • Moving to an end room & winning the game
  • Etc.
Adventure Design – CLI, Revisited

"go North"

Console Input

"You are in the Lagoon. Items: Sword"

Console Output

Your Adventure Game Engine

- currentRoom: Lagoon
- inventory: []
- mode: Normal
Your Adventure Game Engine

```
{"commandName": "go", "commandValue": "North"}
```

Adventure Service

```
{"error": false, "id": 0, "message": "You are in the Lagoon. Items: Sword", "imageUrl": "http://...", "videoUrl": "youtube.com/..."
...}
```

Your Adventure Game Engine

- `currentRoom`: Lagoon
- `inventory`: []
- `mode`: Normal
The **Command Class**

private String commandName;
The keyword of the command. E.g.: "go", "take", "unlock", etc.

private String commandValue;
The argument to the command. E.g.: "North", "sweatshirt", "big door"

private String playerName;
An unused field for this semester; please ignore
Command Example

• Whereas before, your program would accept: "go up and out"

• ...your API receives

  
  {
    "commandName": "go",
    "commandValue": "up and out"
  
  }
The **GameStatus** class

```java
private boolean error;
True if an unexpected error came up; the
private int id;
The numerical ID for the instance that issued the command
private String message;
The message to display to the user, including info about the room.
private String imageUrl;
A publicly accessible URL to an image to show on the page
```
The GameStatus class, cont.

private String videoUrl;
A YouTube video link to play sound from

private AdventureState state;
An object with key-value pairs to show in a table on the website

private Map<String, List<String>> commandOptions;
A map where each key is a possible command name, and the list value represents all the possible string arguments to that command
Example of `commandOptions`:

```
{  
  "go": ["North", "West", "South"],  
  "take": ["Sword"]  
}
```
Visit the pinned thread on Campuswire to ask your questions
If you see a question you'd like to hear the answer to, upvote it