



Interactive Example: Degree Maker



Creates Custom Degrees
for Universities

What is Degree Maker?

- Degree Maker creates degrees
 - Any size
 - Two orientations
 - Include images
 - Include text using any Font
 - Can preview and print
 - Etc.

Use Case

- Two large companies make degrees for the entire world
- They typically have 100 people creating degrees at the same time
- They want a product that allows their employees to make degrees on their computers simultaneously, and store them in a product's repository

Model

- DegreeManager Class
 - Degree newDegree()
 - freeDegree(Degree)
 - preview(Degree)
 - print(Degree, numCopies)
 - saveDegreeToRepository(Degree)
 - Degree loadDegreeFromRepository(Identifier)
 - deleteDegreeFromRepository(Identifier)
 - Identifier[] getRepositoryIdentifiers()

Model (2)

■ Degree Class

- setSize(width in inches, length in inches)
- setOrientationPortrait(isPortrait)
- addSchoolName(String, Font, fontSize)
- addDegreeName(String, Font, fontSize)
- addRecipientName(String, Font, fontSize)
- setWaterMark(Image)
- addImage(Image, upper-left co-ordinate)
- addText(String, Font, fontSize, upper-left co-ordinate)
- Identifier getIdentifier()
- Corresponding get()s for each set/add.

Specifications

- A degree manager must be created first
- Degrees are created through the degree manager
- Size and orientation of degree must be set before any other degree method
- Any of the other methods on the degree are allowed in any order
- Memory is only limitation to number of degrees that are in use at any given time
- One degree manager in charge of one repository (where things get saved)

Types of Testing

- Functional <- Automate
- Unit
- System <- Automate

Functional Testing

- How would you automate functional testing?
 - Using a framework
 - Like JUnit
- Each test case focuses on one function
- Test cases usually grouped by class
- Should have multiple test cases focusing on same function in accordance to complexity
- Covers calling functions with valid and invalid parameters
- Call functions using boundary parameters
- `setup()` and `tearDown()`

System Testing

- Can all of system testing be automated?
 - No, the look of the degree (preview) needs to be analyzed/confirmed by human
- For the automated part, let's use a custom framework

Criteria for SVT Success

- Stress Test
 - No errors during run
 - 200 clients creating/working with degrees
 - Duration: 24 hours

System Test Framework

- What services need to be put into the framework?
 - Logging
 - Test invocation
 - Serial
 - Parallel (concurrency)
 - Reporting
 - Random Object
 - Data
 - Flow (Weightings)

Test Modules

- Start DegreeManager
- Stop DegreeManager
- Create a Degree
- Free a Degree
- Preview a Degree
- Print a Degree
- Save degree
- Load random degree
- Delete random degree
- Set random size
- Set random orientation
- Add random school name
- Add random degree name
- Add random recipient name
- Set random watermark
- Add random image
- Add random text