EECS 448 Team 14

Project 4 Integration Strategy

Our team used a GitHub repository to combine our code from the different features that we were implementing. If members were working on the same feature, then we programmed separate functions within the same file in order to minimize and avoid merge conflicts. This also follows the principle of minimal coupling and high cohesion since the functions and methods do not depend on one another to function properly. We only pushed changes to the main branch since we felt none of the changes were large enough to require their own separate branch, although we still acknowledge that it is considered best practice to have different branches for each of the main game features. Since the project timescale was only a little over 2 weeks, we managed with just using the main branch and had very few merge conflicts as a result of the methods and practices mentioned above.

We used the Sandwich Integration strategy with a greater emphasis on Bottom-Up. First we used Top-Down to implement the larger abstract classes first such as the gameplay loop for our Deal or No Deal game. Once those were complete, we used Bottom-Up Integration to code the specific gameplay features, such as choosing briefcases to remove and accepting or rejecting the Banker's offer. Following our Project 3 prototype we also added the option to play the game according to US or UK rules. Most of the individual game features were integrating using Bottom-UP because we wanted to make sure any single gameplay option, such as swapping cases, was up and running independently before we tried incorporating it into the main gameplay loop. We felt this helped us avoid more complicated errors than if we had tried the Top-Down approach for the gameplay features.