Mark Faust (mfaust4)

Gateway Computing: Java

Greg Anderson

9 December 2022

Object-Oriented Programming Writeup

In this final project, we effectively leveraged the principles of object-oriented programming which makes our code both scalable and simpler. By employing things like encapsulation, composition, inheritance, and polymorphism, this project can reproduce the popular game of Crazy Eights.

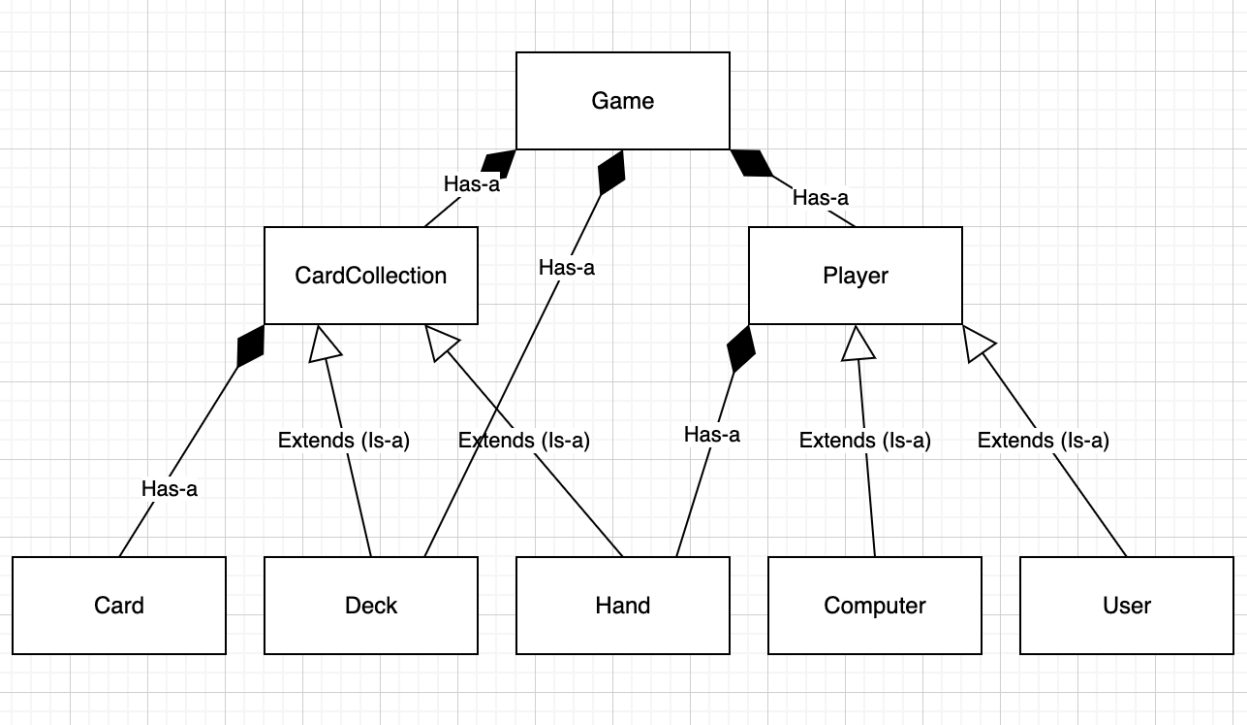
What makes object-oriented code useful is the ability to use abstract concepts of real-life things in code. This is accomplished by encapsulating the attributes of these things into code classes where they can then be accessed and mutated using programmer-defined methods. For example, a standard Player of Crazy Eights would have their attributes of their name and the cards they hold encapsulated into a class of type Player.

Having its own attributes in a class makes Player a new object that can be used. Since a Player's attributes is another object, a Hand with its own set of Cards, we can see the concept of composition in action. Oftentimes, programs can be incredibly complex, so programmers can use other objects as the attributes of a separate object, as seen with the Player's Hand attribute.

However, not all objects are related to each other in the same way. Objects can exemplify has-a and is-a relationships. For example, a User is a Player, and a Player has a Hand. As in the case of User being a Player, we say that User "inherits" all the attributes of the Player class. Inheritance is helpful when wanting to create a new Player object but with some of its own unique attributes/abilities. Unique functionality wanted to be given to a human player (User) so that they could make their own, unautomated move in the game while still having a name and Hand.

Finally, the principle of polymorphism can be seen in the Game class's array of Player objects. Polymorphism helps programmers group objects with similar attributes together in arrays. Since the Computer and User classes inherit attributes of the Player class, both Computers and Users can be gathered into an array that can hold Player objects. This is allowed because Computers and Users have is-a relationships with Player. Without polymorphism, Players of different types would not be able to play Crazy Eights together.

The implementation of this program was significantly simplified by employing the proven object-oriented programming practices.


Game
Has-a
Has-a
Has-a
CardCollection
Player
Has-a
Extends (Is-a)
Extends (Is-a)
Has-a
Extends (Is-a)
Extends (Is-a)
Card
Deck
Hand
Computer
User