Implementation

Demented Characters (G14, C6)

We chose to use flags inside our Character class and its subclasses to determine whether a Character is demented or not. This made sense as the Character class is a common ancestor to both Player and Mob. Another option might have been to have a different superclass, say DementedCharacter, and a new set of subclasses from this to contain the logic for our demented characters. We chose not to follow this architecture as we wanted our characters to be able to transition between the demented state and without. It would also have meant that a lot of our logic would have to be written in two places, due to Java's lack of multiple inheritance, which we also wanted to avoid.

Flagging and unflagging demented Characters (G14, G14.2, C6, C6.1)

To allow demented mobs to attack and damage sane mobs (G14.1), the Filter on the Dynamic Body for all characters was updated in their respective classes. A new entity group was created, represented by PhysicsEntity.DEMENTED_BITS, and the Filter definitions updated such that Players and Mobs are mutual enemies as well as enemies of demented Mobs. When a mob enters or leaves the demented state, their filter is updated to allow for correct collision within a specialized overridden method.

Logic for effects of demented status on Players (C6)

Players are affected differently by the demented status than Mobs (contrast C6 with G14.1), so this more specialized logic is included in the Player class.
While the player is affected by the demented effect, every few seconds, the player will move in the opposite direction to the inputted commands.
To implement this, two timers were added to the Player class, Player.dementedEffectTime and Player.dementedBetweenEffectTime. These respectively measure the amount of time that the player has been acting against the input and the amount of time that the player has been respecting the input. Inside the update() method, there is a call to the new updateDementedTimers() method, which changes the timers and inverts input at the correct times.

Logic for effects of demented status on Mobs (G14, G14.1)

While demented, mobs will switch between one of several unusual behaviours randomly, which are signified by a nested enum located inside Mob. One of these behaviours, DementedMobBehaviour.ATTACK_CLOSEST, means demented mobs will attempt to attack the closest character to them, including other mobs. For this to work, the previous PathfindingAI had to be changed to allow for pathfinding to other Characters, achieved by creating a target variable, and using that in place of the previous reference to the player.
The update() method in Mob was updated to separate any logic to be taken by a demented Mob behind an if block. As this logic is shared by all the different subtypes of Mob, we included this logic inside the Mob class.

Cheats (G15, G15.1, G15.2)

To implement the cheats, we chose to use a pair of flags located within the Session class, which is nested within the DuckGame class. This class is easily accessible from both the menu screens and within the actual game, and the flags allow us to easily enable logic within the Player class to enable or disable the cheats.