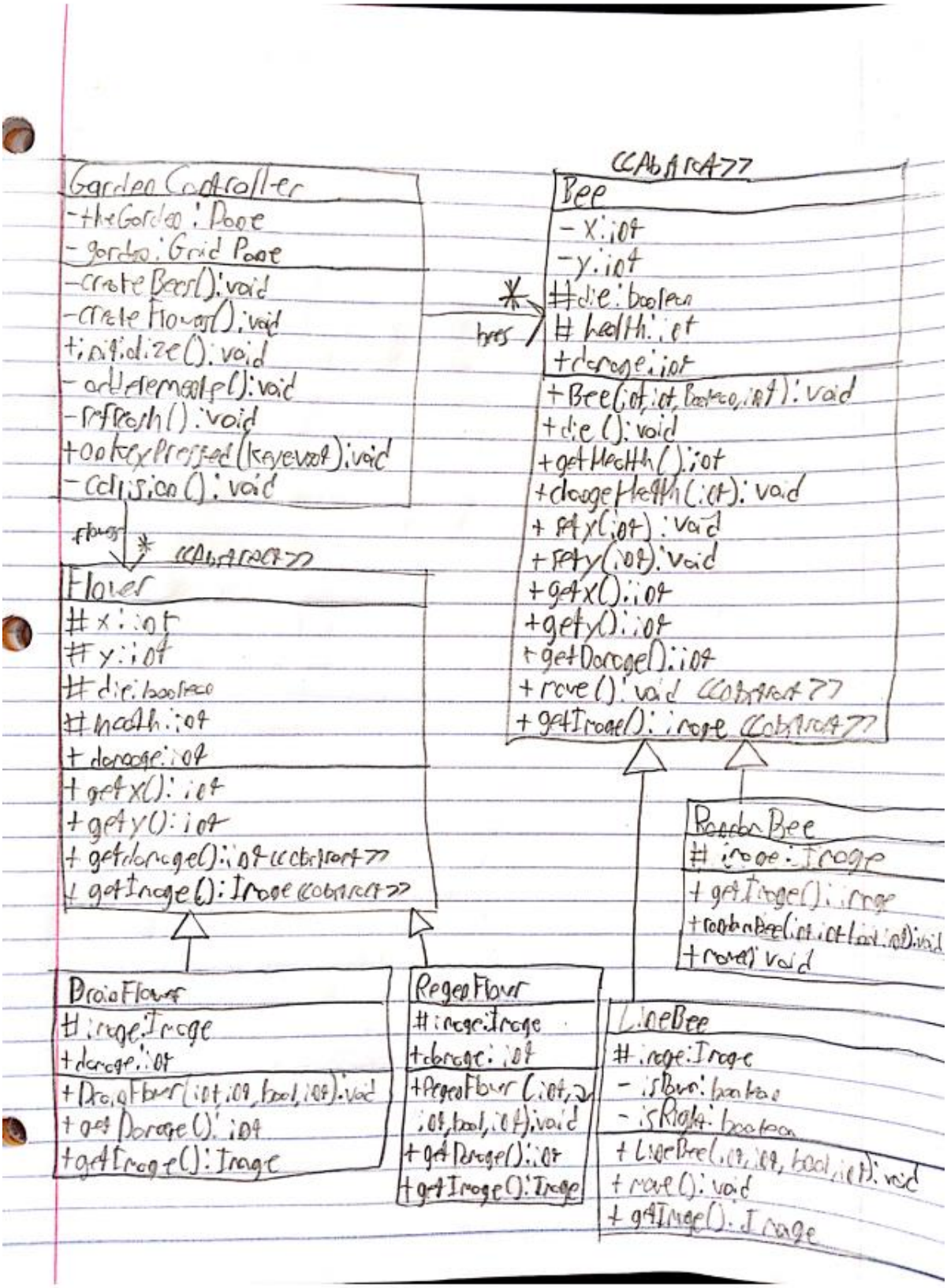
No major changes were needed or made when implementing the code. Methods were added to many of the classes and attributes were added to the Bee and Flower sub and parent classes. We had the Bee and Flower parent and subclasses with minimal methods within them, along with the Controller, main and fxml classes.

Start by understanding the relationships between the classes and how they interact with each other. Start with either the flower or bee abstract class. Think about how will bee interact with flower and its subclasses. And how flower will interact with bee and its subclasses. Once those relationships are defined, move onto how those interactions will play out in the bigger picture (the garden).

When designing the program, do not worry about making all the methods immediately. When implementing the code, the actions caused by the interactions and relationships of the bees, flowers and garden will pop up and will be easy to implement of the bee and flower classes have a good design.

Aidan, Marco, and Jerry created and developed the bee and flower classes. Aidan and Marco worked on the gardenController.

An issue with our program is that the health displayed can be obstructed by other images on the pane when the bees are moving around the garden. Another issue in our program is that when the bees move, the pane will occasionally shift or move.


GardenController
-theGarden: Pane
-garden: GridPane
-createBees(): void
-createFlowers(): void
+initialize(): void
-addRemoveEl(): void
-refresh(): void
+onKeyPressed(KeyEvent): void
-collision(): void
flowers
* <<Abstract>>
Flower
#x: int
#y: int
#dir: boolean
#health: int
+damage: int
+getX(): int
+getY(): int
+getDamage(): int <<abstract>>
+getImage(): Image <<abstract>>
<<Abstract>>
Bee
-x: int
-y: int
#die: boolean
#health: int
+damage: int
bees
+Bee(int, int, boolean, int): void
+die(): void
+getHealth(): int
+changeHealth(int): void
+setX(int): void
+setY(int): void
+getX(): int
+getY(): int
+getDamage(): int
+move(): void <<abstract>>
+getImage(): image <<abstract>>
RobberBee
#image: Image
+getImage(): image
+robberBee(int, int, int, int): void
+move(): void
DroidFlower
#image: Image
+damage: int
+DroidFlower(int, int, bool, int): void
+getDamage(): int
+getImage(): Image
RegenFlower
#image: Image
+damage: int
+RegenFlower(int, int, int, bool, int): void
+getDamage(): int
+getImage(): Image
LineBee
#image: Image
-isBorn: boolean
-isRight: boolean
+LineBee(int, int, bool, int): void
+move(): void
+getImage(): Image