Updated UML Diagram:

n/a

Significant changes:

No new classes were added; however, the way in which Bees and Flowers interact was established. In our implementation, both Bees and Flowers have specified "interact" methods – for Bees, there exists an interact-with-Bee method and an interact-with-Flower method, while Flowers just have an interact-with-Bee method since they generally do not move around. This allows all interactions and movements to be done entirely within the Garden class. The "vector" object (which is just a double array with size 2) was added to dictate movement; whenever a bee needs to move in a different direction, the vector object is updated rather than constantly directly calculating and updating the position.

Advice to next year's class:

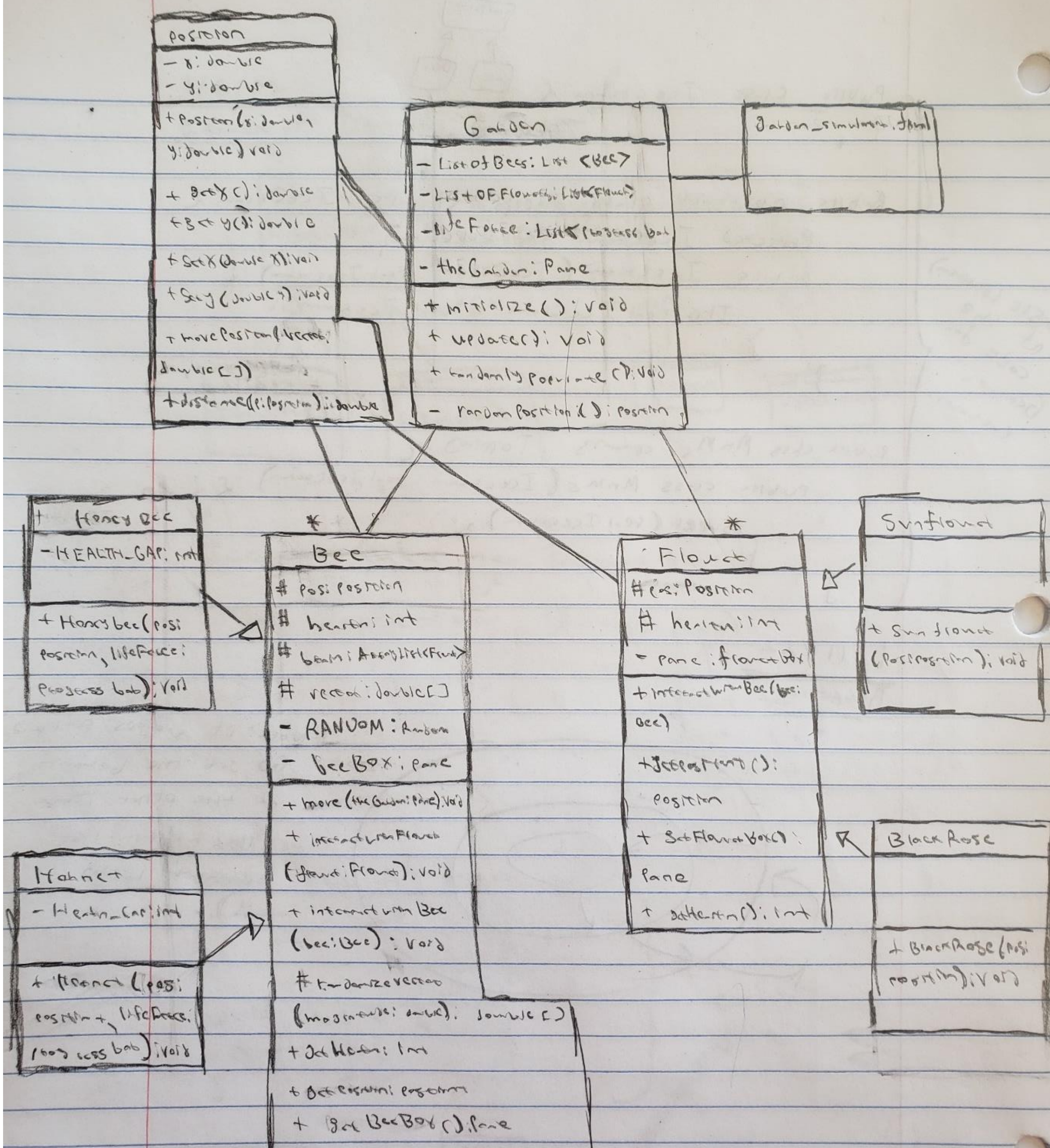
To make the lab easier, make sure you apply the Strategy pattern to streamline different types of Bees and Flowers, as having the base abstract class for both types of objects makes creating variations on the type of object much easier and a lot more rigid. Additionally, it allows simplification of storage as you would not need to include a list for each variation of object; rather you only need to make a list for each type as the type encapsulates all variations.

The spots to cut back on are in Bee behavior. I thought it would be wise to try and implement more complicated movements such as having Bees remember where found flowers are and fly to them, but at the end I scrapped the idea entirely and just decided on random movement. The vector implementation remained though, as it streamlined the movement process a lot.

How the code was split:

Sean did all the Bee and Flower objects (along with all of their variations, such as Sunflowers and Hornets), and also added class comments, and also wrote this PDF.

Terry created the Garden class and also setup the (very complicated) FXML and JavaFX interactions so that you can actually see our project in motion.


position
- x: double
- y: double
+ position(x: double, y: double) void
+ getX(): double
+ getY(): double
+ setX(double x): void
+ setY(double y): void
+ movePosition(vector: double[])
+ distance(p: position): double
Garden
- ListOfBees: List <Bee>
- ListOfFlowers: List<Flower>
- lifeForce: List<ProgressBar>
- theGarden: Pane
+ initialize(): void
+ update(): void
+ randomlyPopulate(): void
- randomPosition(): position
garden_simulator.fxml
HoneyBee
- HEALTH_GAP: int
+ HoneyBee(pos: position, lifeForce: ProgressBar): void
*
Bee
pos: position
health: int
beam: ArrayList<Flower>
vector: double[]
- RANDOM: Random
- beeBox: Pane
+ move(theGarden: Pane): void
+ interactWithFlower(flower: Flower): void
+ interactWithBee(bee: Bee): void
# randomizeVector(magnitude: double): double[]
+ getHealth: int
+ getPosition: position
+ getBeeBox(): Pane
+ setUp(Double x, Double y): void
Hornet
- Health_Cap: int
+ Hornet(pos: position, lifeForce: ProgressBar): void
*
Flower
pos: Position
health: int
- pane: flowerBox
+ interactWithBee(bee: Bee)
+ getPosition(): position
+ setFlowerBox(): Pane
+ getHealth(): int
Sunflower
+ Sunflower(pos: position): void
BlackRose
+ BlackRose(pos: position): void