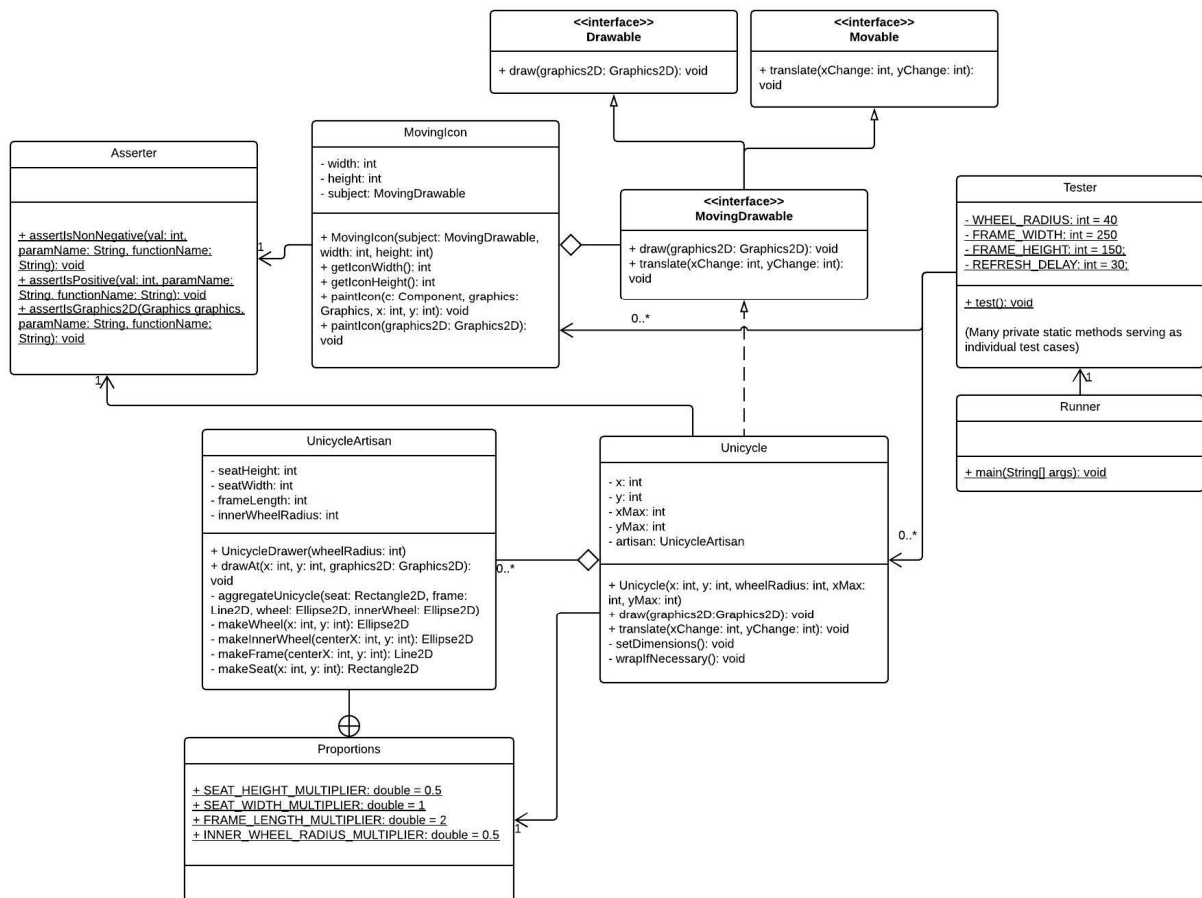**java.awt.geom.Point2D**, which both define abstract getters that return doubles[13][14]. **Dimension2D** and **Point2D** are part of Java Graphics2D API, which is a second version of the Java Graphics API released earlier[15]. **Dimension** and **Point** give public access to their int fields so that the classes can both support the legacy Java Graphics API as well as the newer Java Graphics2D API with implementation of the getters that return doubles.

1.6 UML

I completed all 3 steps of the programming part. Here is the UML after completing step 1:

Here is the UML after completing step 2:

---

[13] https://docs.oracle.com/javase/7/docs/api/java/awt/geom/Dimension2D.html

[14] https://docs.oracle.com/javase/7/docs/api/java/awt/geom/Point2D.html#getX()

[15] http://www.ntu.edu.sg/home/ehchua/programming/java/j8b_game_2dgraphics.html

<<interface>>
Drawable

+ draw(graphics2D: Graphics2D): void

<<interface>>
Movable

+ translate(xChange: int, yChange: int): void

Asserter

+ assertIsNonNegative(val: int, paramName: String, functionName: String): void
+ assertIsPositive(val: int, paramName: String, functionName: String): void
+ assertIsGraphics2D(Graphics graphics, paramName: String, functionName: String): void

MovingIcon

- width: int
- height: int
- subject: MovingDrawable

+ MovingIcon(subject: MovingDrawable, width: int, height: int)
+ getIconWidth(): int
+ getIconHeight(): int
+ paintIcon(c: Component, graphics: Graphics, x: int, y: int): void
+ paintIcon(graphics2D: Graphics2D): void

<<interface>>
MovingDrawable

+ draw(graphics2D: Graphics2D): void
+ translate(xChange: int, yChange: int): void
+ getMaxX(): int
+ getMaxY(): int

Runner

+ main(String[] args): void

Tester

- WHEEL_RADIUS: int = 40
- FRAME_WIDTH: int = 250
- FRAME_HEIGHT: int = 150;
- REFRESH_DELAY: int = 30;

+ test(): void

(Many private static methods serving as individual test cases)

UnicycleArtisan

- seatHeight: int
- seatWidth: int
- frameLength: int
- innerWheelRadius: int

+ UnicycleDrawer(wheelRadius: int)
+ drawAt(x: int, y: int, graphics2D: Graphics2D): void
- aggregateUnicycle(seat: Rectangle2D, frame: Line2D, wheel: Ellipse2D, innerWheel: Ellipse2D)
- makeWheel(x: int, y: int): Ellipse2D
- makeInnerWheel(centerX: int, y: int): Ellipse2D
- makeFrame(centerX: int, y: int): Line2D
- makeSeat(x: int, y: int): Rectangle2D
- setDimensions(): void

Unicycle

- x: int
- y: int
- xMax: int
- yMax: int
- artisan: UnicycleArtisan

+ Unicycle(x: int, y: int, wheelRadius: int, xMax: int, yMax: int)
+ draw(graphics2D:Graphics2D): void
+ translate(xChange: int, yChange: int): void
+ getMaxX(): int
+ getMaxY(): int
- wrapIfNecessary(): void

Race

- competitors: MovingDrawable[]
- REFRESH_DELAY: int = 30
- MAX_VELOCITY: int = 30
- MIN_VELOCITY: int
- MIN_RANDOM_SIZE_MULTIPLIER: double = 0.5
- MAX_RANDOM_SIZE_MULTIPLIER: double = 2

+ Race(competitors: MovingDrawable[])
+ raceRandomCompetitors(numCompetitors: int, normalWheelRadius: int, fraceWidth: int): Race
+ launch(initialVelocity: int): void

Proportions

+ SEAT_HEIGHT_MULTIPLIER: double = 0.5
+ SEAT_WIDTH_MULTIPLIER: double = 1
+ FRAME_LENGTH_MULTIPLIER: double = 2
+ INNER_WHEEL_RADIUS_MULTIPLIER: double = 0.5
+ TOTAL_HEIGHT_MULTIPLIER: double

VelocityManager

- velocity: int
- movingDrawables: MovingDrawable[]
- labels: JLabel[]

+ VelocityManager(initialVelocity: int, movingDrawable: MovingDrawable[], labels: JLabel[])
+ actionPerformed(e: ActionEvent): void
+ statesChanged(e: ChangeEvent): void

Here is the UML after completing step 3:

<<interface>> Drawable

+ draw(graphics2D: Graphics2D): void

<<interface>> Movable

+ translate(xChange: int, yChange: int): void

UnicyclePack

- velocityMultiplier: double
- children: List<Unicycle>

+ UnicyclePack(velocityMultiplier: double)
+ getVelocityMultiplier(): double
+ setVelocityMultiplier(val: double): void

+ draw(graphics2D: Graphics2D): void
+ translate(xChange: int, yChange: int): void
+ getMaxX(): int
+ getMaxY(): int
+ add(child: MovingDrawable): void
+ remove(child: MovingDrawable): void
+ getChildren(): MovingDrawable[]

<<interface>> Pack

+ draw(graphics2D: Graphics2D): void
+ translate(xChange: int, yChange: int): void
+ getMaxX(): int
+ getMaxY(): int
+ add(child: MovingDrawable): void
+ remove(child: MovingDrawable): void
+ getChildren(): MovingDrawable[]

<<interface>> MovingDrawable

+ draw(graphics2D: Graphics2D): void
+ translate(xChange: int, yChange: int): void
+ getMaxX(): int
+ getMaxY(): int

Asserter

+ assertIsNonNegative(val: int, paramName: String, functionName: String): void
+ assertIsPositive(val: int, paramName: String, functionName: String): void
+ assertIsGraphics2D(graphics: Graphics, paramName: String, functionName: String): void

MovingIcon

- width: int
- height: int
- subject: MovingDrawable

+ MovingIcon(subject: MovingDrawable, width: int, height: int)
+ getIconWidth(): int
+ getIconHeight(): int
+ paintIcon(c: Component, graphics: Graphics, x: int, y: int): void
+ paintIcon(graphics2D: Graphics2D): void

Runner

+ main(String[] args): void

Tester

- WHEEL_RADIUS: int = 40
- FRAME_WIDTH: int = 250
- FRAME_HEIGHT: int = 150;
- REFRESH_DELAY: int = 30;

+ test(): void

(Many private static methods serving as individual test cases)

UnicycleArtisan

- seatHeight: int
- seatWidth: int
- frameLength: int
- innerWheelRadius: int

+ UnicycleDrawer(wheelRadius: int)
+ drawAt(x: int, y: int, graphics2D: Graphics2D): void
- aggregateUnicycle(seat: Rectangle2D, frame: Line2D, wheel: Ellipse2D, innerWheel: Ellipse2D)
- makeWheel(x: int, y: int): Ellipse2D
- makeInnerWheel(centerX: int, y: int): Ellipse2D
- makeFrame(centerX: int, y: int): Line2D
- makeSeat(x: int, y: int): Rectangle2D
- setDimensions(): void

Unicycle

- x: int
- y: int
- xMax: int
- yMax: int
- artisan: UnicycleArtisan

+ Unicycle(x: int, y: int, wheelRadius: int, xMax: int, yMax: int)
+ draw(graphics2D: Graphics2D): void
+ translate(xChange: int, yChange: int): void
+ getMaxX(): int
+ getMaxY(): int
- wrapIfNecessary(): void

Race

- competitors: MovingDrawable[]
- REFRESH_DELAY: int = 30
- MAX_VELOCITY: int = 30
- MIN_VELOCITY: int
- MIN_RANDOM_SIZE_MULTIPLIER: double = 0.5
- MAX_RANDOM_SIZE_MULTIPLIER: double = 2

+ Race(competitors: MovingDrawable[])
+ raceRandomCompetitors(numCompetitors: int, normalWheelRadius: int, fraceWidth: int): Race
+ launch(initialVelocity: int): void

Proportions

+ SEAT_HEIGHT_MULTIPLIER: double = 0.5
+ SEAT_WIDTH_MULTIPLIER: double = 1
+ FRAME_LENGTH_MULTIPLIER: double = 2
+ INNER_WHEEL_RADIUS_MULTIPLIER: double = 0.5
+ TOTAL_HEIGHT_MULTIPLIER: double

VelocityManager

- velocity: int
- movingDrawables: MovingDrawable[]
- labels: JLabel[]

+ VelocityManager(initialVelocity: int, movingDrawable: MovingDrawable[], labels: JLabel[])
+ actionPerformed(e: ActionEvent): void
+ statesChanged(e: ChangeEvent): void