CST 239 Activity1 UML Diagram Patrick Buhmbi 2/3/2024

1) **UML class diagram of the Person class.**

The UML diagram for the Person class gives a clear picture of the traits and actions linked with the Person class. Let's break it down:

Class Header: At the top, you see the class name, which is "Person" here. It's like a template for making Person objects in the program.

Attributes Section: Underneath the class name, you find a list of characteristics or features of the Person class. Each one is marked with a hyphen (-), showing they're private, only for use within the class. These features include:

- firstName: String
- lastName: String
- age: int
- weight: double They represent details like a person's first name, last name, age, and weight.

Methods Section: After the attributes, you'll see a list of actions related to the Person class. Each action has a plus sign (+), indicating they are public and can be used from outside the class. The actions include:

- firstName(): String
- lastName(): String
- age(): int
- weight(): double
- setFirstName(firstName: String): void
- setLastName(lastName: String): void
- setAge(age: int): void
- setWeight(weight: double): void These actions let you do things with a person object, like getting their first name or updating it. In short, the UML diagram of the Person class is like a map showing the makeup and abilities of a person in the program. It helps programmers see what details they can store about a person and what actions they can perform with those details. This diagram makes it easier for developers to create, understand, and use the Person class effectively in their Java projects.

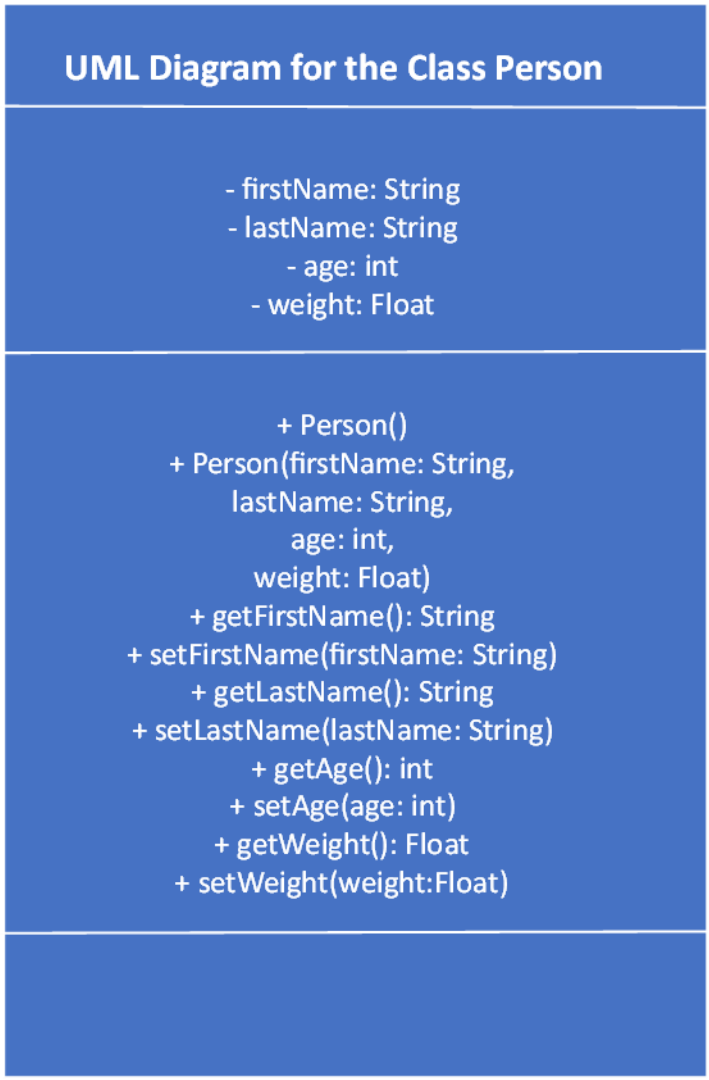

Figure 1 shows the UML diagram for a Person class.

2) UML class diagram of a Race Car Game

The UML diagram for the Car Race Game classes provides a clear overview of the characteristics and behaviors associated with each class. Let's simplify it:

Engine Class:

Attributes:

- **running**: boolean - This indicates whether the engine is currently running (true) or not (false).

Methods:

- There aren't any methods specified in the diagram, but we can assume the Engine class includes methods for starting, stopping, and restarting the engine.

Tires Class:

Attributes:

- pressure: int - This represents the pressure of the tires.

Methods:

- checkPressure(): This method checks the pressure of the tires.

Car Class:

Attributes:

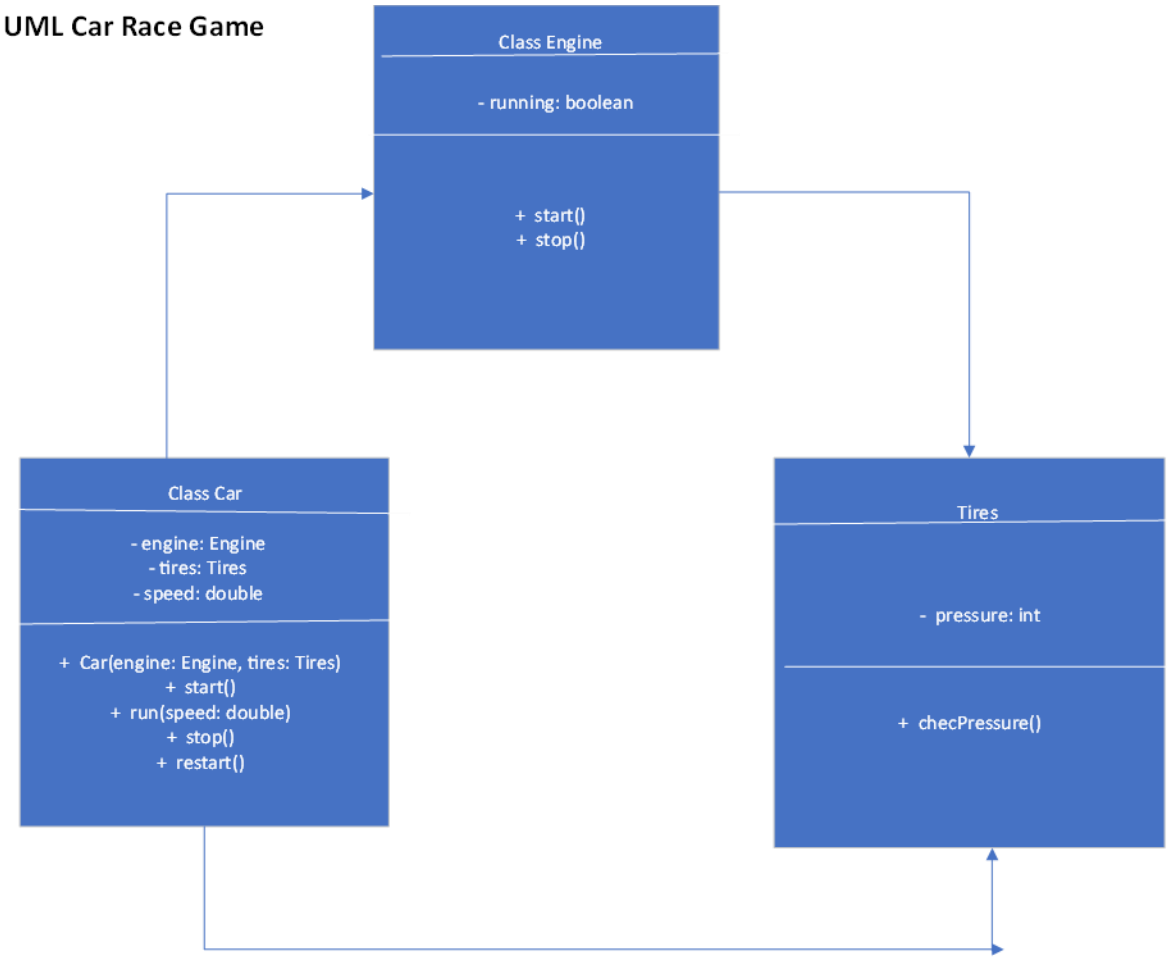

Figure 2 UML class diagram of your complete model.

Car Class:

Attributes:

- engine: Engine - This is the car's engine, represented as an Engine object.

- tires: Tires - These are the car's tires, represented as a Tires object.
- speed: double - This indicates the speed of the car.

Constructor:

- Car(engine: Engine, tires: Tires): This initializes a Car object with a specific engine and tires.

Methods:

- start(): Begins the car's engine.
- run(speed: double): Drives the car at a specified speed.
- stop(): Halts the car's movement.

restart(): Restarts the car's engine

Relationships:

- The Car class has a composition relationship with the Engine and Tires classes, meaning it contains objects of these classes as its attributes.
- The Car class utilizes the Engine and Tires classes to execute various functions like starting, running, stopping, and restarting the car.
- While there aren't explicit relationships displayed between the Engine and Tires classes, they are integral to the Car's functionality, working together within the Car class to manage the car's operations and monitor its status.