

FORMAL METHODS OF SOFTWARE ENGINEERING



**SOFTWARE SPECIFICATION DOCUMENT FOR SELF DRIVING CAR**

Maham Tariq Khan SE-21004

Hamna Aamir SE-21016 Batch: 2021

Year: Third

Dept: Software Engineering

Software Engineering Department

NED University of Engineering and Technology, Karachi.

# OBJECTIVE:

The self-driving car system aims to provide an autonomous driving experience, allowing users to specify and control the vehicle's speed through a user-friendly interface. The system ensures safe and efficient speed adjustments, taking into account user inputs, real-time sensor data, and physical control mechanisms.

# SCOPE:

The scope of the self-driving car system encompasses the development and implementation of an autonomous driving solution with a primary focus on speed control. The scope extends to the communication with physical controls, such as the accelerator and brake, to effectuate the desired speed changes. However, advanced autonomous features beyond speed control, detailed navigation capabilities, and considerations for environmental conditions (such as weather and traffic) are explicitly identified as out of scope. The scope is defined to create a clear understanding of the system's goals, features, and limitations, ensuring alignment with user expectations and project objectives.

## Speed Control:

* + - User can set and modify the desired speed through the user interface.
    - The system adjusts the vehicle's speed based on user inputs and real-time data from sensors.
    - Speed control includes acceleration, deceleration, and maintaining a constant speed.

## Safety Features:

* + - The system ensures that the requested and actual speeds are within predefined safety limits.
    - Real-time monitoring of the vehicle's environment through sensors to make informed speed adjustments.

## User Interface:

* + - A user-friendly interface allows users to interact with and control the self- driving car.
    - Users receive feedback on the system's actions and the current state of the vehicle.

## Physical Controls:

* + - The system communicates with physical controls, such as the accelerator and brake, to implement speed adjustments.

# ASSUMPTIONS:

1. The self-driving car operates in a legal and regulated environment.
2. Users have a basic understanding of the system's capabilities and limitations.
3. System assumes a standard vehicle configuration with compatible physical controls.

# CONSTRAINTS:

1. The system operates within legal speed limits and safety regulations.
2. Hardware limitations and sensor accuracy may impact the precision of speed adjustments.
3. **PROBLEM STATEMENT:**

A self-driving car is a highly critical system requiring significant risk analysis to prevent failures. The failures in such case would result in a loss of both life and property. Therefore, one of the aspects that needs to be monitored and controlled is the speed of the vehicle. For this purpose, this system is designed. This system would enable user to specify the speed of the self-driving vehicle. Initially, the system would check whether the requested speed lies within the specified range of the vehicle and then send a signal accordingly. If the current speed is higher than the requested speed, the system would signal the vehicle to decrease its speed. If the current speed is lower that the requested speed, the system would signal the vehicle to increase its speed. In case of equality of both speeds, the system would be signaled to do nothing and continue.

1. **FUNCTIONALITIES OF THE SYSTEM:**

The functionalities of a self-driving car system with speed control components can be categorized into several key areas

# User Interface:

* **Set Speed:** Allow the user to specify the desired speed for the self-driving car.
* **User Feedback:** Provide feedback to the user about the current speed, system status, and any relevant notifications.

# Speed Control Component:

* **Receive User Input:** Accept the speed input from the user interface component.
* **Check Speed Range:** Verify whether the requested speed is within the acceptable range for the vehicle.
* **Coordinate Acceleration and Deceleration:** Determine whether to accelerate, decelerate, or maintain the current speed based on the user input and current vehicle speed.
* **Send Commands to Physical Controls:** Interface with the physical controls to adjust the vehicle's speed.

# Physical Controls:

* **Accelerator Control:** Receive signals from the speed control component to adjust the vehicle's acceleration.
* **Brake Control:** Receive signals from the speed control component to adjust the vehicle's deceleration.

# Speed Sensor:

o **Monitor Current Speed:** Continuously measure and report the current speed of the self-driving vehicle.

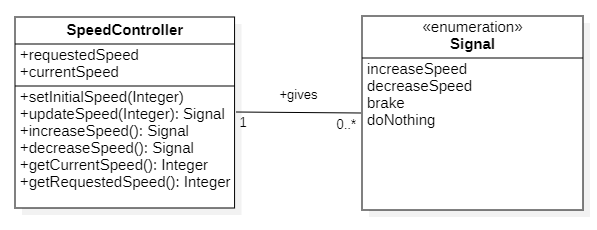
It's important to note that the specific functionalities are based on the sophistication of the self- driving system, the level of autonomy, and the technology used.

# 4+1 ARCHITECTURAL VIEW OF THE SYSTEM:

* 1. **LOGICAL VIEW:**

# 7.1.1. CLASS DIAGRAM:

The class diagram for our self-driving car system provides a concise representation of the key classes and their relationships, offering a blueprint for the system's structure. The diagram includes essential classes such as "SpeedController" and "Signal.".

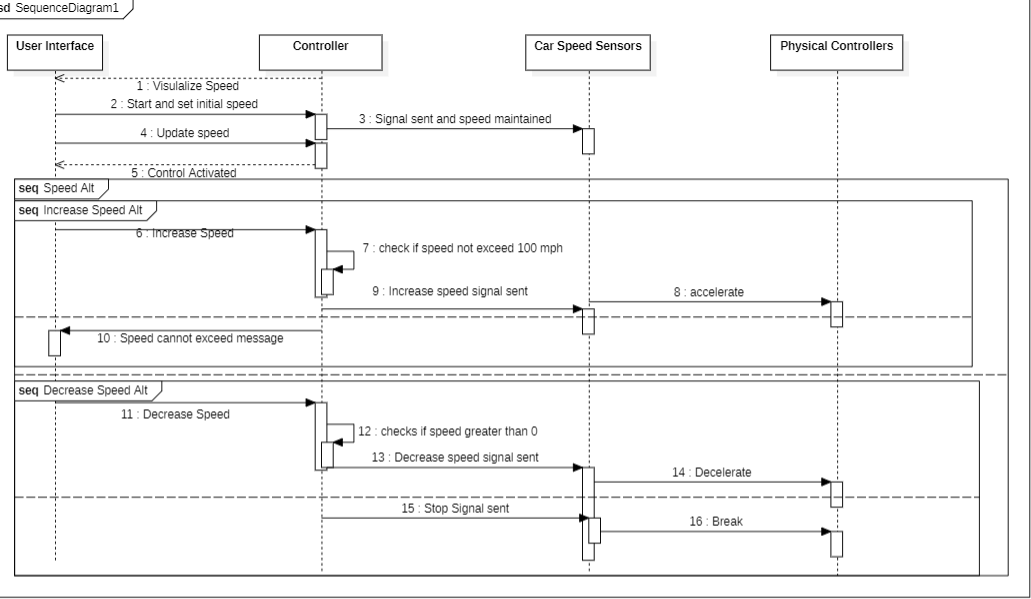


# EXPLANATION:

*Figure 1: Class Diagram of the system*

The "SelfDrivingCar" class encapsulates the overall functionality of the vehicle, including the core components responsible for speed control. It communicates with the "SpeedController" class to relay user preferences and system status.

The "SpeedController" class acts as the brain of the system, coordinating the acceleration and deceleration processes. It interacts with the "Signal" class, receiving real-time speed data for decision-making.



The "Signal" enum focuses reporting the vehicle's current speed, providing crucial input signal to the "SpeedControl" class.

# PROCESS VIEW:

**7.2.1. SEQUENCE DIAGRAM:**

The sequence diagram for our self-driving car system captures the dynamic interactions and message exchanges between three key entities: "UserInterface," "SpeedController," and "Sensors", “PhysicalControllers”. This diagram illustrates the sequence of events as the user sets a desired speed, triggering the speed adjustment process within the system.

*Figure 2: Sequence Diagram of the system*

# EXPLANATION:

The sequence begins when the "UserInterface" sends a message to the "SpeedController" to set the desired speed. This represents the user input through the interface.

The "SpeedController" processes the desired speed, checking if it falls within an acceptable range, ensuring it aligns with the system's constraints and safety measures.

To make informed decisions, the "SpeedController" communicates with the "Sensors" to obtain real-time information about the current speed of the self-driving car.

Based on the user input and current speed obtained from the sensors, the "SpeedController" decides whether to accelerate, decelerate, or maintain the current speed.

The "SpeedController" sends signals to the "PhysicalControls," representing the accelerator, to adjust the vehicle's acceleration accordingly.

Similarly, the "SpeedController" sends signals to the "PhysicalControls," representing the brake, to adjust the vehicle's deceleration when needed.

Finally, the "SpeedController" provides feedback to the "UserInterface" regarding the adjusted speed and the overall system status, ensuring a closed-loop communication with the user.

This sequence illustrates the dynamic interactions among the "UserInterface," “SpeedController," "Sensors," and "PhysicalControls," showcasing the collaborative process involved in adjusting the self-driving car's speed based on user input and real-time sensor data.

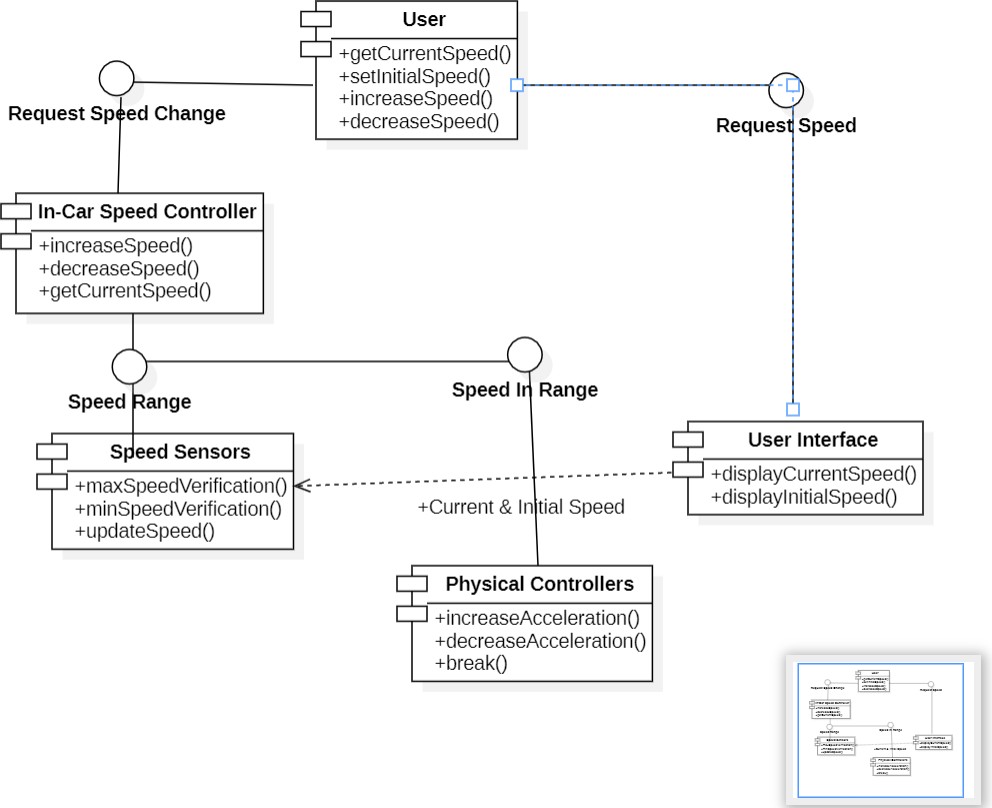
# DEVELOPMENT VIEW:

**7.3.1. COMPONENT DIAGRAM:**

This component diagram illustrates the modular structure of the self-driving car system, showcasing the clear separation of concerns among different components. The user interface handles user interactions, the speed control component manages the logic for adjusting speed, sensors monitor the environment, and physical controls execute the necessary adjustments. This modular design promotes scalability, maintainability, and the ease of understanding the system's architecture.

# EXPLANATION:

*Figure 3: Component Diagram of the system*



The central component in the diagram is the "Self-Driving Car System," representing the overarching system that integrates various components to enable autonomous driving.

The "User Interface" component is responsible for interacting with the user. It allows users to set the desired speed, providing an intuitive interface for user engagement.

The "Speed Control Component" acts as the brain of the system, coordinating the adjustment of the vehicle's speed. It receives inputs from the user interface, checks speed ranges, communicates with sensors, and controls the physical components.

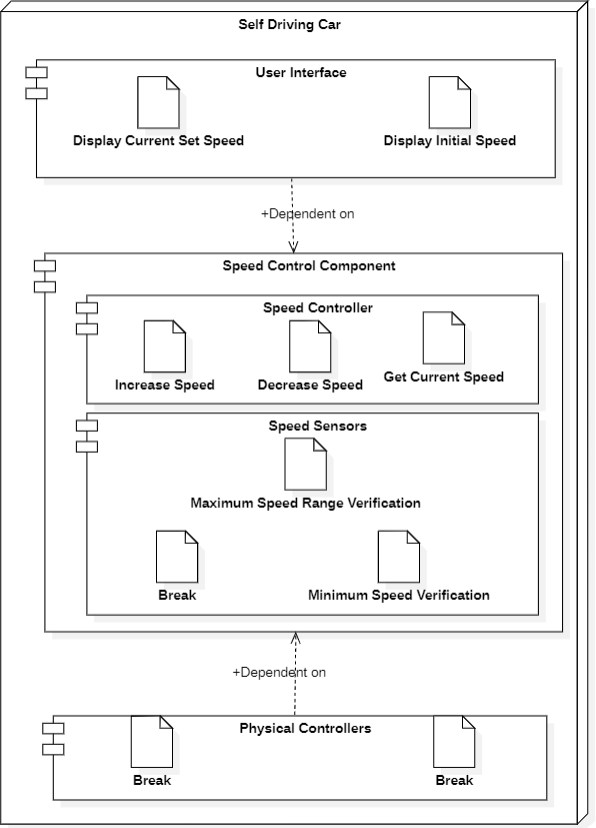
The "Sensors" component represents the suite of sensors responsible for monitoring the vehicle's environment. It continuously gathers data, including the current speed, and provides this information to the speed control component.

The "Physical Controls" component includes the accelerator and brake. It receives signals from the speed control component to adjust acceleration and braking based on the system's decisions.

# PHYSICAL VIEW:

**7.4.1. DEPLOYMENT DIAGRAM:**

This deployment diagram illustrates the physical distribution of components within the self-driving car system. Each node represents a specific computing or hardware environment, and communication channels depict the flow of information between these components. This diagram is crucial for understanding how the system is deployed in a real-world scenario, emphasizing the interactions and dependencies between different components.



*Figure 4: Deployment Diagram of the system*

# EXPLANATION:

The central entity in the deployment diagram is the "Self-Driving Car System," which represents the entire system deployed in a physical environment, such as a vehicle.

The "User Interface Node" represents the hardware or computing environment where the user interface component is deployed. This could be an onboard display or control panel within the vehicle.

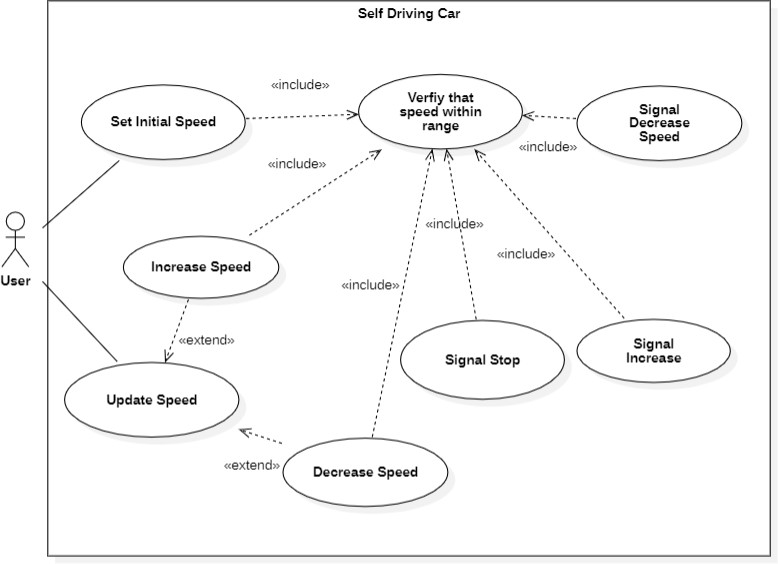
The "Speed Control Node" represents the computing environment where the speed control component is deployed. This node is responsible for processing user inputs, coordinating with sensors, and managing the speed control logic.

The "Sensors Node" represents the physical sensors distributed throughout the self-driving car. These sensors continuously collect data about the vehicle's environment, including the current speed.

The "Physical Controls Node" represents the hardware or computing environment where the physical control components (accelerator and brake) are deployed. This node executes the adjustments based on signals received from the speed control component.

The arrows between nodes represent communication channels. For example, there is a communication link between the "User Interface Node" and the "Speed Control Node" to convey user inputs. Similarly, there are links between the "Speed Control Node" and the "Sensors Node" for receiving real-time data and between the "Speed Control Node" and the "Physical Controls Node" for sending control signals.

# 7.5 USE CASE OF THE SYSTEM:



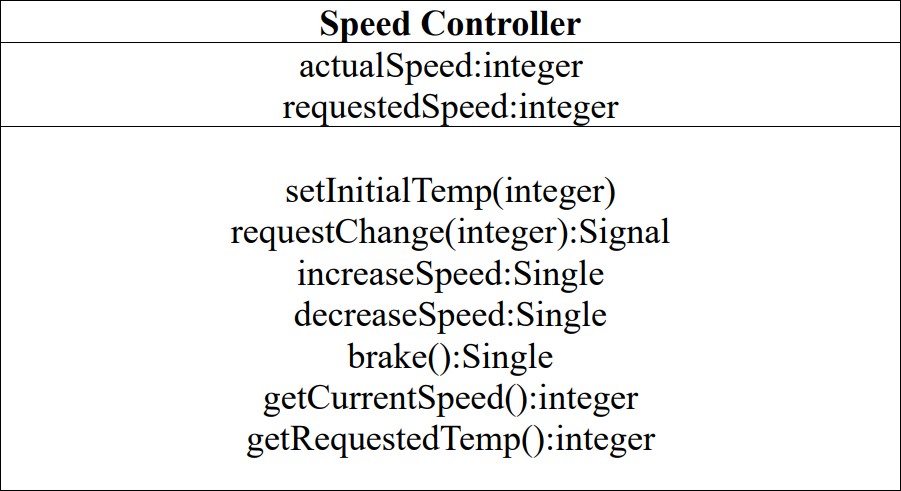
*Figure 5: Use Case Diagram of the system*

# EXPLANATION:

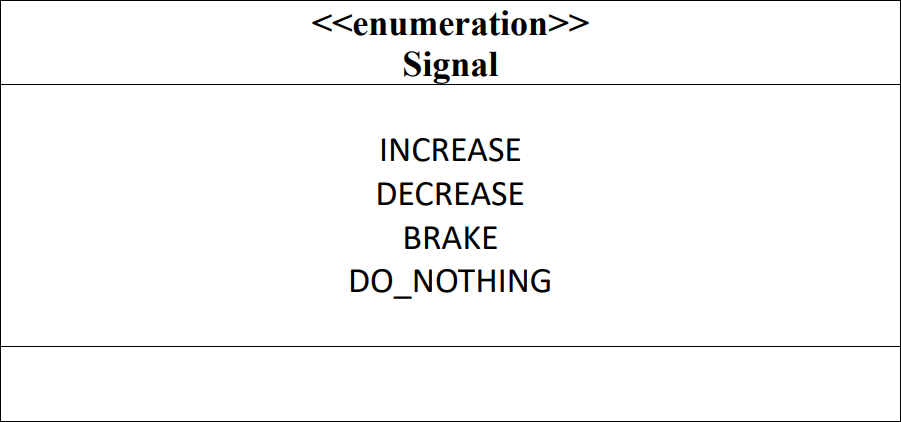
The self-driving car system facilitates user-controlled speed adjustments through a user- friendly interface, empowering users to set their desired speed. Leveraging real -time sensor data, the system ensures safe acceleration, deceleration, or maintenance of speed within predefined safety limits. Users receive feedback, and the system communicates with physical controls for seamless implementation of speed changes. The primary use case revolves around delivering an autonomous and safe driving experience while prioritizing user control over the vehicle's speed, ultimately enhancing both user engagement and safety in the driving process.

# VDM – SL SPECIFICATIONS FOR SPEED CONTROLLER SYSTEM OF SELF DRIVING CAR

* 1. **VDM SPECIFICATION:**



*Figure 5: Specification of the speed controller system*



*Figure 6: UML Specification of the Signal type*

# VDM SPECIFICATIONS:

-- Define the type for signals

## types

Signal = <INCREASE> | <DECREASE> | <BRAKE> | <DO\_NOTHING>;

-- Define constants for speed range

## values

MAX: Z = 100; MIN: Z = 0;

-- Define the state for SpeedController

**state** SpeedController **of** requestedSpeed : [Z] actualSpeed : [Z]

-- actual speed must be in range and requested speed must be in range or equal to nil

**inv mk**\_SpeedController (requestedSpeed, actualSpeed) ∆ (inRange(requestedSpeed) V requestedSpeed = **nil**) ^ (inRange(actualSpeed));

-- both requested actual an requested speed are undefined when system is initialized

**init mk**\_SpeedController (requestedSpeed,actualSpeed) ∆ requestedSpeed = **nil** ^ actualSpeed = **0)**;

## end functions

inRange (val:Z) result:bool

**pre** TRUE

**post** result MIN≤ val≤ MAX

--An operation that records the initial speed of the system

## operations

setInitialSpeed()

**ext wr** speedIn : Z ==> ()

**pre** (speedIn) == (requestedSpeed = **nil** and actualSpeed = 0) and inRange(speedIn)

**post** actualSpeed = speedIn;

--A request change function to change the speed requestChange(speedIn:Z) SignalOut:Signal

**ext wr** requestedSpeed: [Z]

**rd** actualSpeed: [Z]

**pre** inRange (speedIn) ^ (actualSpeed≠ **nil post** requestedSpeed=speedIn ^

(speedIn > actualSpeed ^ SignalOut = <INCREASE>

V speedIn < actualSpeed ^ SignalOut = <DECREASE>

V speedIn = actualSpeed ^ SignalOut = <DO\_NOTHING>);

--An increment function to increase the speed Increment() SignalOut:Signal

**ext rd** requestedSpeed: [Z]

**wr** actualSpeed: [Z]

**pre** actualSpeed < requestedSpeed **^**requestedSpeed ≠ **nil post** actualSpeed = actualSpeed+5

^ (actualSpeed < requestedSpeed ^ SignalOut = <INCREASE> V actualSpeed=requestedSpeed^SignalOut=<DO\_NOTHING>);

--A decrement function to decrease the speed decrement() SignalOut:Signal

**ext rd** requestedSpeed: [Z]

**wr** actualSpeed: [Z]

**pre** actualSpeed > requestedSpeed **^** requestedSpeed≠**nil post** actualSpeed = actualSpeed-5

^ (actualSpeed > requestedSpeed ^ SignalOut = <DECREASE> V actualSpeed=requestedSpeed^SignalOut=<DO\_NOTHING>);

--A brake function

brake () SignalOut:Signal

**ext rd** actualSpeed

**rd** requestedSpeed

**pre** (requestedSpeed = 0 and actualSpeed ≠ 0)

**post** actualSpeed = 0 ^ SignalOut=<BRAKE>;

13 | P a g e

getRequestedSpeed() currentRequested : [Z]

**ext rd** requestedSpeed : [Z]

**pre** TRUE

**post** currentRequested = requestedSpeed

getActualSpeed() currentActual : [Z]

**ext rd** actualSpeed : [Z]

**pre** TRUE

**post** currentActual = actualSpeed

**end** SelfDrivingCar

# JAVA CLASSES IMPLEMENTATION:

**SPEED CONTROLLER CLASS**

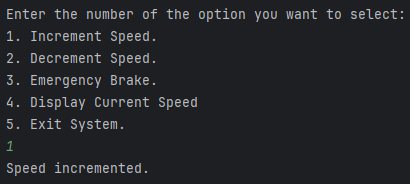
public class SpeedController {  
  
 private Integer requestedSpeed;  
 private Integer actualSpeed;  
 //Assumed the maximum possible speed of vehicle to be 100.  
 public static final int *maximumSpeed* = 100;  
 //Speed of the car can be now lower than 0.  
 public static final int *minimumSpeed* = 0;  
  
 public enum Signal {*INCREASE*, *DECREASE*,*DO\_NOTHING*,*BRAKE*};  
  
 //Initializing the speed of the vehicle.  
 //Requested speed is initially null as no value has been entered yet.  
 //Actual Speed is initially 0 as the vehicle will be initially on rest.  
 public SpeedController() {  
 this.requestedSpeed = null;  
 this.actualSpeed = 0;  
 }  
  
 // Function to check if a value is in the specified range  
 private boolean inRange(int speedVal) {  
 return (speedVal >= *minimumSpeed* && speedVal <= *maximumSpeed*);  
 }  
  
 // Invariant method  
 private boolean invariant() {  
 return (this.requestedSpeed == null || inRange(this.requestedSpeed)) && (inRange(actualSpeed));  
 }  
 public Signal Increment() {  
 this.actualSpeed += 5;  
 if(this.invariant()) {  
 return(Signal.*INCREASE*);  
  
 }  
 else{  
 this.actualSpeed-=5;  
 return(Signal.*DO\_NOTHING*);  
 }  
 }  
 public Signal Decrement() {  
 this.actualSpeed -= 5;  
 if(this.invariant()){  
 return(Signal.*DECREASE*);  
  
 }  
 else{  
 this.actualSpeed+=5;  
 return(Signal.*DO\_NOTHING*);  
 }  
 }  
 public int getCurrentSpeed() {  
 return (this.actualSpeed);  
 }  
 public int getRequestedSpeed() {  
 return (this.requestedSpeed);  
 }  
 public Signal Brake(){  
 if(this.getCurrentSpeed()==0){  
 return(Signal.*DO\_NOTHING*);  
 }  
 else{  
 this.actualSpeed=0;  
 return(Signal.*BRAKE*);  
 }  
 }  
 public Signal requestChange(int SpeedIn) {  
 Signal signalOut = Signal.*DO\_NOTHING*;  
  
 if (inRange(SpeedIn)) {  
 this.requestedSpeed = SpeedIn;  
  
// set appropriate value for output variable  
 if (SpeedIn > this.actualSpeed) {  
 signalOut = Signal.*INCREASE*;  
 }  
 if (SpeedIn < this.actualSpeed) {  
 signalOut = Signal.*DECREASE*;  
 }  
 if(SpeedIn==0){  
 signalOut=Signal.*BRAKE*;  
 }  
 return signalOut;  
 }  
 else {  
 System.*out*.print("Speed is out of range.Enter speed within range 0 - 100 MPH.\n");  
 return (signalOut);  
 }  
 }  
}

**SPEEDCONTROLLER TESTING CLASS**

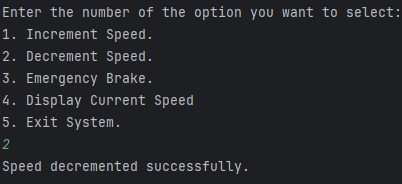
import java.util.\*;  
  
public class SpeedControllerTestingClass {  
 public static SpeedController *newObj*=new SpeedController();  
 public static void promptEnter(){  
 System.*out*.print("\nPress \"ENTER\" to continue...");  
 Scanner scanner = new Scanner(System.*in*);  
 scanner.nextLine();  
 }  
 public static void main(String[] args){  
 System.*out*.print("\*\*\*\*\*STARTING THE SYSTEM\*\*\*\*\*\*\n\n");  
 int sentinel=1;  
 while(sentinel!=5){  
 System.*out*.print("\nEnter the number of the option you want to select:\n1. Increment Speed." +  
 " \n2. Decrement Speed.\n3. Emergency Brake.\n4. Display Current Speed \n5. Exit System.\n");  
 Scanner sc= new Scanner(System.*in*);  
 sentinel= sc.nextInt();  
 switch (sentinel){  
 case 1: {  
 SpeedController.Signal checkIncrement=*newObj*.Increment();  
 if(checkIncrement== SpeedController.Signal.*INCREASE*){  
 System.*out*.print("Speed Incremented.\n");  
 }  
 else{  
 System.*out*.print("Maximum Speed Reached. Cannot Increment Further.\n");  
 }  
 break;  
 }  
 case 2:{  
 SpeedController.Signal checkDecrement=*newObj*.Decrement();  
 if(checkDecrement== SpeedController.Signal.*DECREASE*){  
 System.*out*.print("Speed Decremented.\n");  
 }  
 else{  
 System.*out*.print("Vehicle already at rest. Cannot decrement further.\n");  
 }  
 break;  
 }  
 case 3:{  
 SpeedController.Signal checkBrake=*newObj*.Brake();  
 if(checkBrake== SpeedController.Signal.*BRAKE*){  
 System.*out*.print("Brake was applied. Stopping the vehicle...\n");  
 }  
 else{  
 System.*out*.print("Vehicle already on rest. No application of brakes required.\n");  
 }  
 break;  
 }  
 case 4:{  
 System.*out*.print("The Current Speed is "+*newObj*.getCurrentSpeed()+" MPH.\n");  
 break;}  
 case 5: {  
 sentinel=5;  
 break;}  
 default:{  
 System.*out*.print("Invalid Choice.Select Valid Option\n");  
 *promptEnter*();  
 }} }  
 System.*out*.print("Exiting the system.");  
 }  
}

**CODE OUTCOMES:**

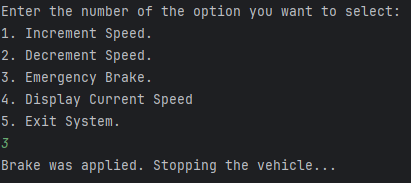
**Increment Speed**

****

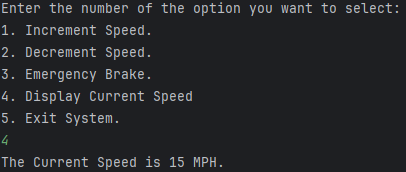
**Decrement Speed**

****

**Emergency Brake:**

****

**Display Current Speed**

****

**10. CONCLUSION:**

In conclusion, the development and implementation of the self-driving car system is highly safety critical. By integrating real-time sensor data and adhering to predefined safety limits, the system ensures a safe and efficient driving experience, encompassing acceleration, deceleration, and the maintenance of a consistent speed. The system's primary focus on empowering users through a user-friendly interface to specify and adjust the vehicle's speed underscores its commitment to user engagement and control.

The delineation of clear in-scope elements establishes well-defined boundaries for the system, contributing to a focused development effort. The acknowledgement of constraints, safety regulations, and hardware limitations, underscores a commitment to responsible and realistic implementation. The system's provision of feedback to users about its actions and the vehicle's state enhances transparency and user understanding.

In essence, the self-driving car system represents a significant stride towards achieving autonomy in vehicular control, with a clear focus on user-centric speed management and safety. The defined scope, constraints, and assumptions provide a robust foundation for development, ensuring that the system aligns with objectives, remains adaptable, and addresses user needs effectively.