40205163 – Calum Hamilton

Coursework – Trains!

Introduction and Overview

The coursework objective is to design a train control system in Ada-SPARK, for the 1954 Borst Atomic train. Various task constraints were given, to increase the difficulty and provide some preconditions and postconditions (e.g. a Speed Limit, Control Rod restrictions and Carriage Restrictions).

Approach

I split my program into three files: a train body file, a train specification file, and a main body file.

File	Purpose
Train body file	Interacts with the train record, updating values
Train specification file	Describes the preconditions, postconditions and global variables
Main body file	Overall train controller. Uses functions described in the body file, with restrictions outlined in the specification file

I have my train set up as a record, as opposed to a list of global variables. This provides some extra protection, accessing the record rather than global variables.

Controller Structure & Descriptions

Train Record Structure:

Element	Data Type	Constraints
Carriage_No	Integer	Range 0 - 5
Carriages	Pre-defined String	Value: (Attached / Unattached)
Electricity	Float	Range 1.0 - 300.0
Emergency_Stopped	Pre-defined String	Value: (Running / Stopped)
MaxElectricity	Float	Range 1.0 - 300.0
MaxSpeed	Integer	Range 0 - 400
Overheating	Pre-defined String	Value: (Overheating / Normal)
Reactor	Pre-defined String	Value: (Online / Offline)
Rod_No	Integer	Range 1 - 5
Rods	Pre-defined String	Value: (Present / Missing)
Speed	Integer	Range 0 - 400
Temperature	Float	Range 1.0 - 300.0
Water	Integer	Range 0 - 100

I ensured where possible, that there were range constraints for all my values.

Global Variables:

Element	Data Type	Explanation
Maximum Carriages	Integer (Constant)	The maximum number of carriages the train can support
Maximum Rods	Integer (Constant)	The maximum number of rods the train reactor can support
Overheating Temperature	Float (Constant)	The temperature at which the reactor overheats
Speed Limit	Integer (Constant)	The speed limit of the track (not the maximum speed limit of the train)

Function Summary:

Function	Purpose
Accelerate	Accelerates the train
Brake	Decelerates the train
Carriage_Add	Adds a carriage to the train
Carriage_Rem	Removes a carriage from the train
Emergency_Stop	Causes the train to come to a complete stop
Reactor_Offline	Takes the reactor offline
Reactor_Online	Takes the reactor online
Rod_In	Adds a rod to the reactor
Rod_Out	Removes a rod from the reactor
Update_Current_Electricity	Updates the current electricity produced
Update_Max_Electricity	Updates the maximum electricity that can be produced
Update_Max_Speed	Updates the maximum speed the train can reach
Update_Temp	Updates the current temperature
Water_Add	Adds water to the reactor
Water_Rem	Removes water from the reactor

Function Preconditions and Postconditions:

Function	Precondition	Postcondition
Accelerate	• Speed < Max Speed • Speed < Speed Limit • Reactor = Online	• Speed <= Max Speed • Speed <= Speed Limit
Brake	• Speed > 0 • Reactor = Online	• Speed >= 0
Carriage_Add	• Speed = 0 • Carriage_No < 5	• Speed = 0 • Carriage_No <= 5
Carriage_Rem	• Speed = 0 • Carriage_No > 0	• Speed = 0 • Carriage_No >= 0
Emergency_Stop	• Emergency_Stopped = Stopped	• Speed = 0
Reactor_Offline	• Speed = 0 • Reactor = Online	• Speed = 0
Reactor_Online	• Speed = 0 • Reactor = Offline	• Speed = 0
Rod_In	• Speed = 0 • Reactor = Offline • Rod_No < 5	• Rod_No = Rod_No'old + 1 • Rod_No <= 5
Rod_Out	• Speed = 0 • Reactor = Offline • Rod_No > 1	• Rod_No = Rod_No'old - 1 • Rod_No >= 1
Update_Current_Electricity	• Reactor = Online	
Update_Max_Electricity	• Rod_No > 0 • Speed < Max Speed • Speed < Speed Limit	
Update_Max_Speed	• Speed < Speed Limit	
Update_Temp	• Rod_No > 0 • Water <= 100 • Water >= 0	
Water_Add	• Speed = 0 • Water < 100 • Reactor = Offline	• Water <= 100
Water_Rem	• Speed = 0 • Water > 0 • Reactor = Offline	• Water >= 0

Conclusion and Analysis

There are several shortcomings in my application.

Maximum Speed / Maximum Electricity / Water / Control Rods

Water and control rods cannot be changed after the reactor has been turned on. This adds a lot of simplicity to the application as it allows the maximum speed and maximum electricity to be calculated before the train turns on the reactor.
In real-world reactors, rods are added into a during the energy generation, to allow the increase or decrease of temperature and reactions per second. Water would also be able to be added during the moving of the train (at least as an emergency measure to cool down the reactor).

Update Functions

The update functions should have postconditions specifying the range and ensuring the range of the value is not out with the acceptable range. I have restricted the range the functions can be, within the formulae and within the main controller.

Future Work

For future work, the train could run between 2 platforms, with the distance being calculated by a basic speed-to-distance calculation. It also opens the possibility of enabling the carriage doors to open and shut and passengers to enter and exit the train.
For additional future work, there could be more than 1 train on the line, allowing for multiple trains, however interaction between trains on the line could be potentially complex.

Difficulties Overcame

Difference in Language

Ada-SPARK is a very different style of previous Object-Oriented and Functional programming from what I have previously encountered. It took me a lot of time to become confident; even then I struggled with integrating pre-conditions and post-conditions with the running of the application, and ensuring the dependencies were correct.