COS 214 The Racing Simulation Project

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

- **Date issued**: 28 September 2018
- **Date due**: 29 October 2018 at midday
- **Submission procedure**: Upload via the CS website

Table of Contents

1. Introduction

During this project you will make use of at least ten different design patterns, some of them already implemented in previous practicals. Some guidelines are provided on which patterns to use and how to implement them for the project. You are however allowed to replace them with similar patterns if you can provide a proper reason for the change. You may also add additional patterns if needed. In addition to implementing the project code, you are also required to submit Doxygen and project documentation. Please read the entire specification before starting the project.

2. Constraints

1. This project must be completed individually or in pairs. You will be given an opportunity to register your pair. Please make sure you do so, if you do not register your pair beforehand, you will have to work individually.
2. During the project demonstration both members of a pair must be able to fully explain all parts of the project.
3. The project has to compile and run on the Linux system installed on the lab computers.

3. Submission Instructions

You are required to upload all your source files and documentation as a single archive (tar.gz or zip) to the CS website before the final deadline. The archive should contain two PDFs, one for the Doxygen documentation and one for the project documentation with your project description and UML diagrams. No other formats besides PDF are allowed. You do not have to bring a hard copy of the documentation to the project demo.

No late submissions will be accepted. Do not show up during the marking session with your code on a flash drive, since your code from the uploaded archive will be used during the demonstrations.

Additional milestones between 28 September and 22 October will be given with specific requirements for achieving the milestone.

4. The Story

You are an avid racing car follower. It does not matter what type of car it is, as long as it is fast and competing. After spending your weekends at race tracks and rallies, you are known to drivers, crews and management. On numerous such outings you have got to know the manager of the Myalami Race Track. Over the long weekend that has just passed, the Myalami Race Track manager approached you to create a system that can simulate car races.

This document serves to guide you in your efforts to design the simulation system.

5. Pattern Overview

Pattern used	**Combines with / Links with**
Abstract factory	Used to create cars. Combines with Builder.
Decorator	Decorates cars in two ways. Adds behaviour to tracks. Combines with Abstract Factory and Composite.
Prototype	Clones cars with or without their added behaviour. Combines with Abstract Factory and Decorator.
Observer	Pit stop crew observes cars. Combines with Prototype/Decorator/AbstractFactory car products and Mediator.
Mediator	Assigns cars to tracks. Pit stop crew operates on cars. Combines with Prototype/Decorator/AbstractFactory car products and Observer for cars and Decorator and/or Composite for tracks.
Composite	Used to create a track.
State	Current status of the car. Combines with Prototype/Decorator/AbstractFactory car products
Strategy	Adds driving styles to cars. Combines with Prototype/Decorator/AbstractFactory car products
Visitor	Manipulates cars on the track. Combines with Composite/Decorator, Prototype/Decorator/AbstractFactory car products and Strategy.
Iterator	Checks race status on the track. Combines with Decorator/Composite.
Façade	Encapsulates all functionality and combines with classes in the subsystem to provide a management/control interface for running racing simulations.
Builder	Creates cars with differentiated parts.

6. Building cars

6.1. Part One: Building a Car

Before you can begin with any simulations, you require cars. Each car is created based on a preferred model and its use. For instance, you may have different representations of a car: standard, sports (e.g SUV, sedan, wagon) and electronic for different usages (e.g rally, F1, karting, Monster trucks, Road…). These categories are not fixed and are up to you to choose and define. You require at least 3 models and 3 usages. Make use of the Abstract Factory pattern to create different car factories for your preferred models and usages.

Minimum patterns to use:

- Abstract Factory

Milestone: Submit this part on 8 October 2018 at midday and demo on 8 or 9 October 2018.

6.2. Part two: Let's jazz it up

Being satisfied with your current factory you realise there is something missing; you feel that your designs are dull and need some flare. You reach out to one of your friends in the digital art business to design some fancy vinyls for the cars (at least 2). The most appropriate pattern for this functionality is the Decorator.

You also want to add behaviour to the cars without affecting other cars which were created by the same factory. For instance, you may add different colours or abilities, including speed, traction (2 or 4 wheel drive), tyres, steering wheel, etc. to the car. Again, the Decorator is used to provide this functionality. Remember, at any one time a car can have multiple added abilities.

Minimum patterns to use:

- Decorator

Milestone: Submit this part on 8 October 2018 at midday and demo on 8 or 9 October 2018.

6.3. Part three: Get production going

With the cars going into production and publicity surrounding the simulation mounting, Myalami Race Track has seen an increase in interest in the simulation. People are craving the same designs of cars already produced to "buy" and privately race in the simulation. Myalami Race Track expands its business to include a subsidiary called Myalami Simulation Works, headed by you and your team. To improve business prospects, you decide to design a framework that can build your cars either in a unique fashion for the more esteemed client, or as a "copy" of a previously designed car.

All cars produced are cloned using the Prototype design pattern. In some instances, only the base car is cloned without any added behaviour. The other option is to clone a fully designed car - that is the basic car plus its added vinyls and behaviours. The client does not have to know about the concrete implementations and codes only to references.

Minimum patterns to use:

- Prototype

Milestone: Submit this part on 8 October 2018 at midday and demo on 8 or 9 October 2018.

7. Racing cars

7.1. Part four: Getting ready to race

Before long, interested parties who have bought their virtual cars want to know when the track on which these cars are to race will be ready. You appoint a person to manage the registration of cars on the simulation tracks that you are in the process of building. One car may be entered on multiple simulation tracks, naturally they cannot race concurrently. The pattern you decide to use to provide for this functionality is the Mediator.

Minimum patterns to use:

- Mediator

Milestone: Submit this part on 15 October 2018 at midday and demo on 15 or 16 October 2018.

7.2. Part five: Build the track

The simulation race track consists of different sections. By joining these sections together, a track is designed (very much like how scale electric tracks are put together). Examples of sections are: straight, 1/8 turn right, 1/8 turn left, straight with left peel off/on, straight with right peel off/on. Putting 8 right 1/8 turns together, creates a circular track. The Composite pattern can be used to build larger track sections out of the basic available sections, which in turn can be put together to form the entire track.

Use the Decorator pattern to add additional armco (barriers), flags, sand pits and the like to the track sections. Specifically, you require a pit stop and a start/finish line. You must add at least one other type of decorator.

Additionally, the track pieces must be able to keep record of which cars are currently on it. This will be used during the racing part of the simulation in the project.

Minimum patterns to use:

- Composite
- Decorator

Milestone: Submit this part on 15 October 2018 at midday and demo on 15 or 16 October 2018.

7.3 Part six : Getting a pitstop crew ready

Cars are linked to teams, with each team being allocated a pitstop. A pitstop services the cars as they come in, mostly for refuelling or tyre changes. Some cars need to be repaired by the pit crew if they were involved in a bumper bashing on the track. Each pit crew requires at least a manager who decides when a team car is to come in, a refueller and 4 tyre changers.

To simulate an actual pitstop, let certain actions take longer than others. This will result in leaderboard changes during the simulated race.

Minimum patterns to use:

- Observer
- Mediator

Milestone: Submit this part on 15 October 2018 at midday and demo on 15 or 16 October 2018.

7.4 Part seven : Time to race

7.4.1 The race

A race comprises of a number of phases. These phases may include, but are not limited to:

- Test lap
- Race ready
- Race start
- Race in progress
- Race paused
- Race end

A race manager oversees the race, usually from the control tower. Announcing when the race is about to start or end. He is also responsible for displaying each car's position in the race after each lap so that they can see who is first.

A race car also comprises of different internal states. These states may include, but again are not limited to:

- Ready
- Racing
    - Damaged
    - Fuel level
    - Tyre wear
- Stopped

These states must determine whether or not a car has to visit a pit stop or other roadside destinations.

Minimum patterns to use:

- State
- Observer

7.4.2 Traversing the track

You notice, by watching more real-world races, that there are common driving styles among the racers. Some are reckless, some are cautious and others may have a more balanced style. To improve the real-world experience of the simulation, you decide to include driving styles into the simulation. Make use of the Strategy pattern to dynamically assign a driving style to a “racer” of a car. For instance, the racer assigned to a car may decide to go faster but move more recklessly, or go slower and have a lower chance of crashing into opponent vehicles.

Make use of a timer to coordinate the race. On each tick, you can iterate over the track and move vehicles forward according to various properties and the driving strategy of the racer assigned to the car. You must then keep track of when the cars have finished a race in order to determine a winner.

The Visitor and/or Iterator patterns are used to manipulate and monitor the cars on the track. An Iterator can be successfully used to check the state of the track at a particular moment and accordingly update the leaderboard. The Visitor will provide a means of moving the cars along the track. Different visitors can provide different means by which the cars are moved along the track.

Minimum patterns to use:

- Strategy
- Visitor and/or Iterator

Milestone: Submit this part on 22 October 2018 at midday and demo on 22 or 23 October 2018.

8. Putting it all together

8.1 Part eight: One system to rule them all

A successful year has passed and you are exhausted by all your current duties. As a brilliant thinker, you decide that it will be more cost efficient to design a system rather than hiring a group of people to manage all of your **current and possible future** duties.

Use the Façade design pattern to provide a unified interface to all your other systems currently implemented.

Minimum patterns to use:

- Façade

Milestone: Submit this part on 29 October 2018 at midday and demo on 29 or 30 October 2018 along with the entire system.

8.2 Part nine: Building unique cars

Currently, the Abstract Factory builds a very basic car. You want to improve the simulation by including car frames, engines, wheels and more to build different models of cars (with different engines or speed). Use the Builder design pattern to encapsulate the variations and separate it from the stable part of your car model.

Link this building process in with the Abstract Factory of Part one.

Minimum patterns to use:

- Builder

Milestone: Submit this part on 29 October 2018 at midday and demo on 29 or 30 October 2018 along with the entire system.

8.3 Part ten: Milking more money

After seeing the interest people showed in the races, you decided to add the option for viewers to add a bet on their favourite racing car(s). The betting system should allow a viewer to make only one bet per racing car, with a minimum of R500 per bet. Detailed information should be shown to you (as the manager), including but not limited to the total amount of money that viewers currently bet on a race and how much each viewer bet on each racing car.

Use creativity as much as possible to determine how the payouts should work.

Minimum patterns to use:

- Up to you to choose any reasonable pattern(s)

Milestone: Submit this part on 29 October 2018 at midday and demo on 29 or 30 October 2018 along with the entire system.

9. Bonus

Bonus marks will be given for any additional patterns you add to the project up to a maximum of 3 patterns. Note, when adding a pattern you must be able to state why it is a good choice and how you implemented the design pattern. Additionally, a more detailed user interface will more likely receive better marks.

10. Documentation

In addition to the automated Doxygen documentation, you will have to submit an additional PDF containing your project documentation. This should include a general overview of the project, descriptions of the individual design patterns and how/why they were used. If you decide to use different or additional patterns to the ones suggested, you should also document this. Add UML diagrams with Visual Paradigm. You need a class diagram for the entire project and at least one of each of the other UML diagrams covered in the course. For instance, the Observer and Mediator can be nicely represented with a sequence diagram. A state diagram can be used to illustrate the current state of a racing car. You don't have to write many pages of documentation. Taking the diagrams into account, five to ten pages should suffice. Professional and correct documentation will earn more marks.

11. Demonstration

You have to demonstrate your project during the booked sessions. If working in a group, both team members have to be present and should be able to fully explain all aspects of the project. If you do not pitch for the demo session, you will forfeit all marks for the project, even if you uploaded a functional program. You will be required to compile and run your program from the uploaded archive and explain various parts of the project by referring to the submitted code and documentation. The marker will ask you questions about the implemented design patterns and you will have to show and explain the necessary code. Please arrive at least ten minutes prior to your booked time.