Group -2

Railway Platform Allocation System

MC 122: Object Oriented Programming

Group Members:

Shrut Sutariya id: 202103011

Rajnandani Ambasana id: 202103031

Dhruv Goti id: 202103033

Harsh Vaghela id: 202103034

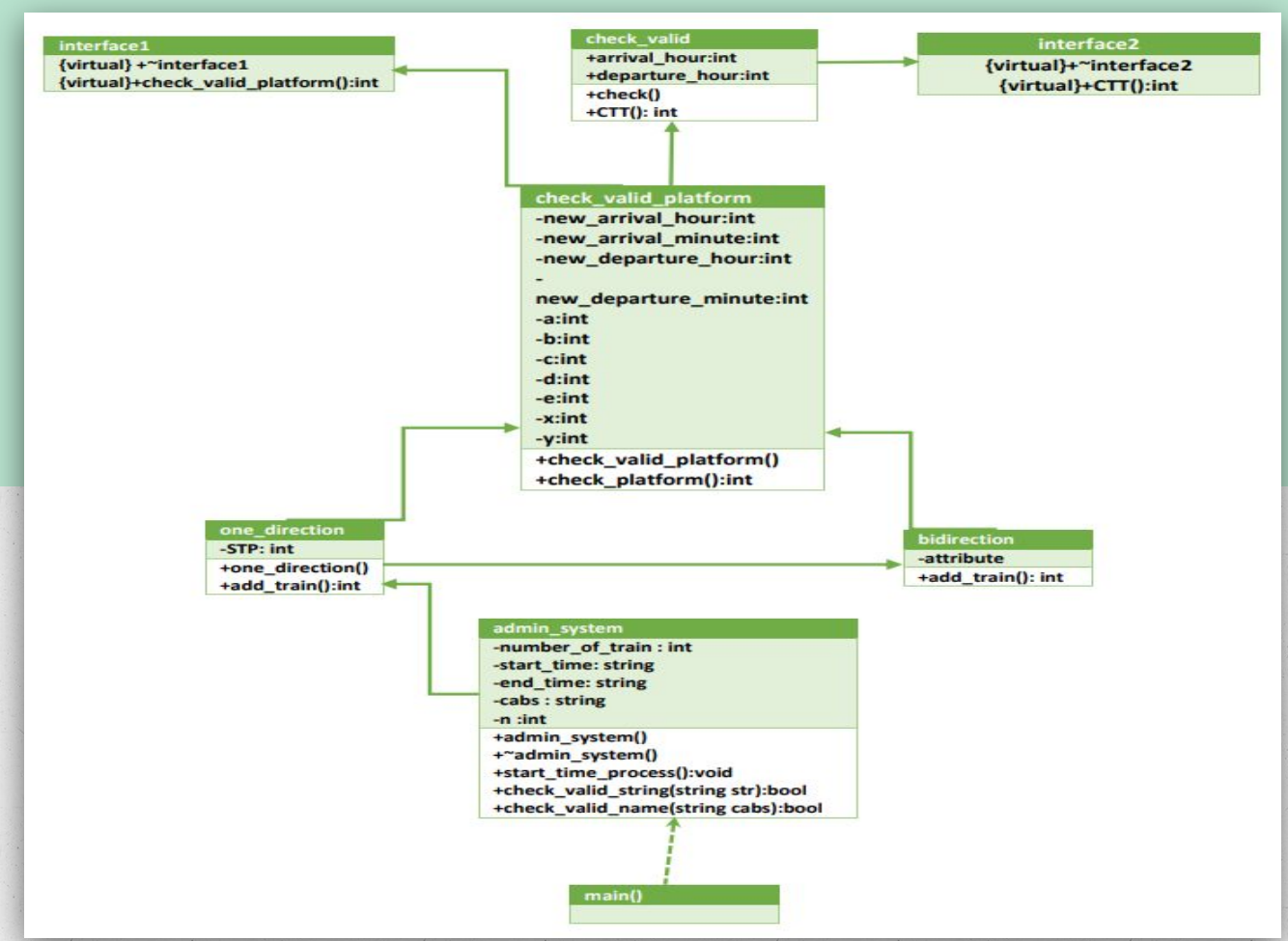
Sanjay kodiyatar id: 202103035

About the project

- Railway is one of the most important system of our country. To effectively manage the timetable and allocate the platform, we need to consider so many factors.
- While allocating the platforms, we need to consider the timings of the trains so they don't face any clashes. For example, if any train comes before the previous train's departure, then we need to allocate that train, another platform.
- Another problem is accommodation. If the length of the train is greater than the platform length, then we cannot allocate that platform to that train. Bidirectional railway line is a system which allows to run the trains in both the directions. Such platforms have capacity to accommodate more than one train on a single line. They contain loops and junction to change their lines during arriving or departure. Advantage is, we can reduce the cost of construction and maintenance. It also require a very accurate management of allocation. We are going to solve such problems in our project work.

Requirements

- Here we will take total number of trains as an input.
- Then we will take time as a string in the 24 hrs formate (hh:mm).
- And finally we will take number of cabins in that train from the user.
- So our program will return the platform number for that train as the result.


interface1
{virtual} +~interface1
{virtual}+check_valid_platform():int
check_valid
+arrival_hour:int
+departure_hour:int
+check()
+CTT(): int
interface2
{virtual}+~interface2
{virtual}+CTT():int
check_valid_platform
-new_arrival_hour:int
-new_arrival_minute:int
-new_departure_hour:int
-
new_departure_minute:int
-a:int
-b:int
-c:int
-d:int
-e:int
-x:int
-y:int
+check_valid_platform()
+check_platform():int
one_direction
-STP: int
+one_direction()
+add_train():int
bidirection
-attribute
+add_train(): int
admin_system
-number_of_train : int
-start_time: string
-end_time: string
-cabs : string
-n :int
+admin_system()
+~admin_system()
+start_time_process():void
+check_valid_string(string str):bool
+check_valid_name(string cabs):bool
main()

Concept of OOP

- ***Constructor :***

Constructor are mainly use in this whole project for assigning default value in the variable. Here we use Default Constructor.

- ***destructor:***

We use destructor for printing thank you at the end of program .

- ***'this' keyword :***

'this' keyword is used to refer to the object which invokes the method. 'this' keyword helps to use same variable name for both local and instance variables.

- ***Inheritance :***

we use simple inheritance : A child class derives member variables and methods from only one parent class.
->check_valid_platform inherit from check_valid.

- ***Multilevel Inheritance :***

A child class derives member variables and methods from another derived class.
->Here One_direction inherit from check_valid_platform.
->admin_system inherit from One_direction.

- ***Multiple Inheritance :***

A child class derives member variables and methods from multiple parent classes.
->one_direction inherite from ckeck_valid_platform and bidirection

- ***hierarchical Inheritance :***

one base class is derived by many child classes.
->check_valid_platform derive two child classes - one_direction and bidirection.

- ***Interface :***

in interface1 we declare method name check_platform. Implement this method in class name check_valid_platform
in interface2 we declare method name CTT. Impliments this method in class name check_valid.

- ***Thread method :***

sleep()
->Here we use sleep() for stop executing program for 1sec .

- ***Exception handling :***

if user enter wrong time or wrong no of cabs then it will throw the exception .

- ***access modifier ()***

->we use different type of modifier like public and private .

- ***Other :***

stoi()

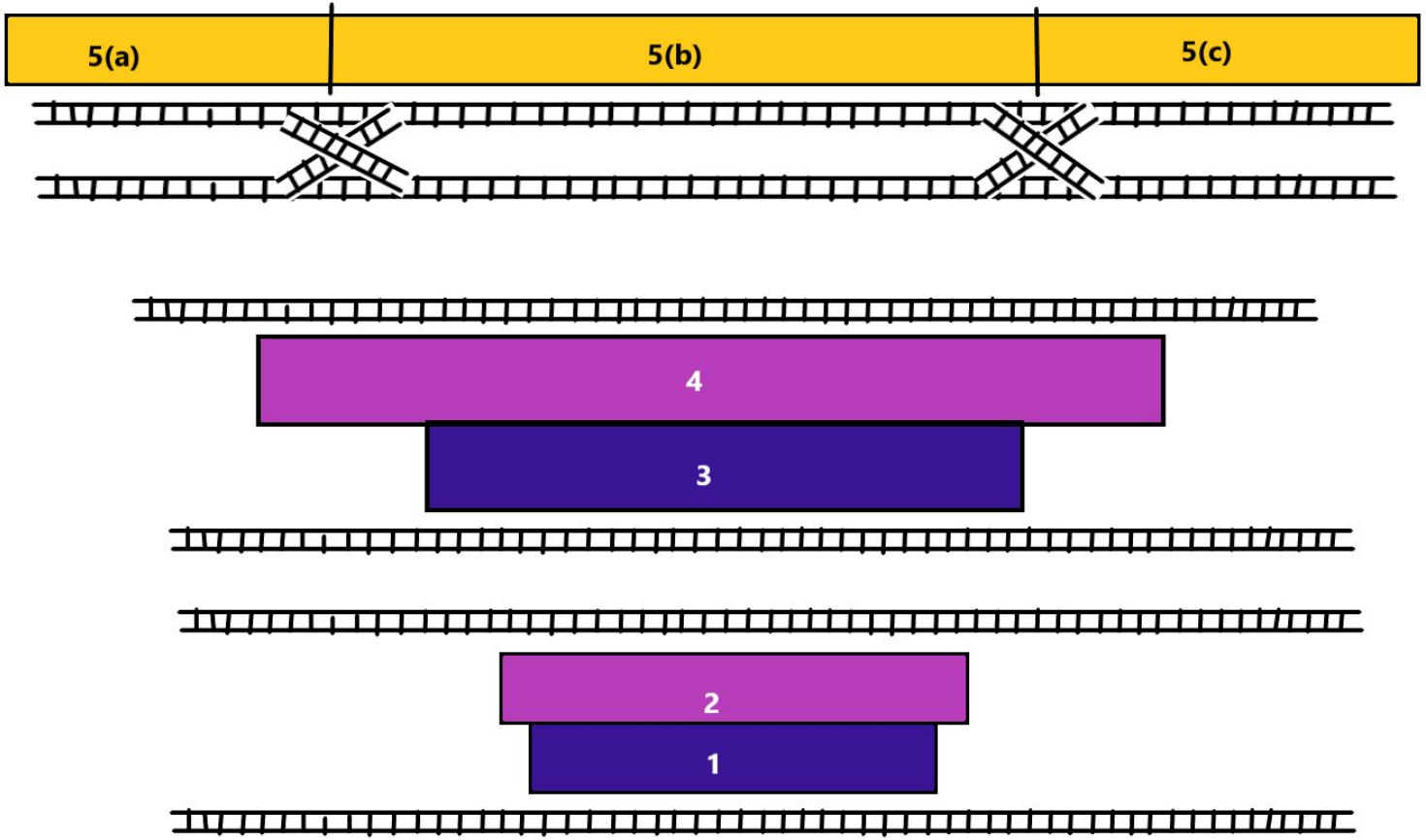
->we use stoi() for converting string to integer.

Code Explain :

- In This Code We Have Assume That Only 5 Platforms Are Here.
  - First Four Platform Are Single Direction Platform.
  - And The Capacity Of Each Platform For occupied Train's Cabs Is

Platform	Capacity
1	15
2	20
3	25
4	30
5	40

  - Our Fifth Platform Are Bidirectional Platform.
  - In This Platform We Occupied 3 Section Means In Same Time We Can Arrange 3 Train In This Platform.
  - If Admin Of System Enter The Number Of Cabs Greater Than 40 Then We Can Not Occupied That Train In Our Railway Platform Allocation Project.


5(a)
5(b)
5(c)
4
3
2
1

Demonstration :

Click here to see the code

Lesson Learnt and understanding

- Through this project, we could learn the use of so many concepts of OOP. We could analyse the problem of railway allocations.
- We needed to think very hard to implement each and every test cases, so it really helped to improve our logic and thinking skills.
- By using the inheritance and polymorphism, we could use the methods very efficiently rather than module programming.
- Keywords made the variables easy to access.
- Exception handling made our program more organised and helped to debug.

Contribution

- ***Coding:***
  Dhruv Goti
  Sanjay Kodiyatar
  Shrut Sutariya

- ***Code rectification and ppt:***
  Rajnandani Ambasana
  Harsh Vaghela

Thank you