Lab 1 Transpotting

Group 3: Muzhe Zhou, Ruohan Li

How to run

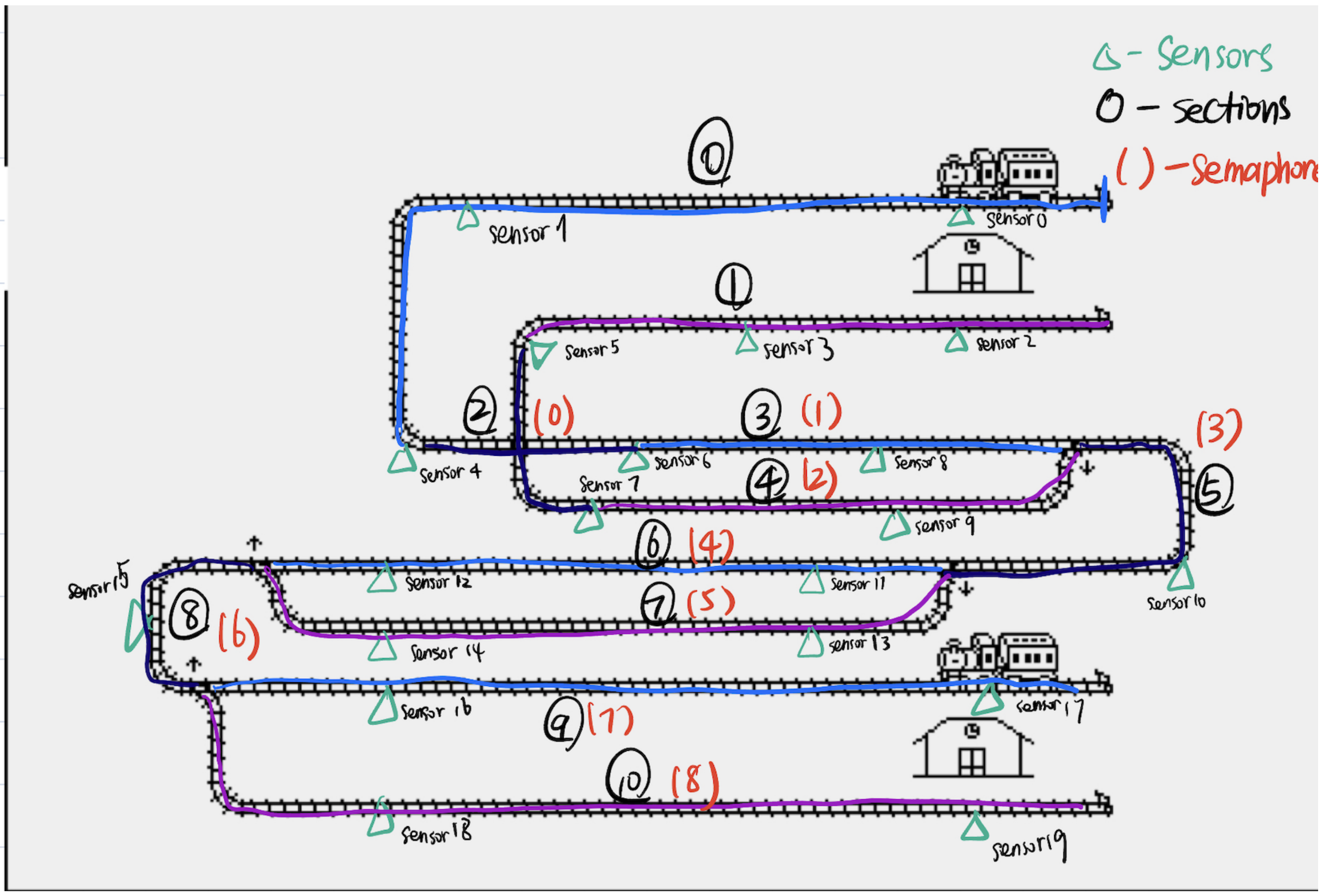
```
make
```

```
java -cp bin Main Lab1.map 5 20
```

`5` and `20` are speeds for the two train.

Placement of the sensors

We have chosen 20 sensors in total to avoid derailment and collision. There are four sensors close to the stops used for stopping the train. In Section `0` and `1`, two sensors are placed for detecting if a train is going to leave or has entered this section. There are four sensors close to the crossing part in section `2`. They are used for detecting if a train has entered this crossing area. In Section `3`, `4`, `5`, `8`, there is one sensor each to detect if a train pass this section. There are two sensors in Section `6` and `7`, one for release the semaphore in the last section and one for acquire the next semaphore. Since this is an essential area, two sensors make the traffic more efficient. In Section `9` and `10`, there are two more sensors for detecting if a train is going to leave or has entered this section.



Choice of critical sections

The section 5 and 8 are critical because only one train can occupy these tracks at a time.

Section 3 and 4, Sections 6 and 7, and Sections 9 and 10 are critical as switches must be set for different directions to prevent derailment and collisions.

Section 2 though has two separate rails can lead to collision.

Maximum train speed

We set the maximum speed value is 20. If the speed is too high, the train may not decelerate to zero in time, potentially causing it to cross into the track of another moving train, leading to a collision, or crash into the end of the platform when stopping at the station.

How you tested your solution

We wrote a Python script to test the program's performance under different speed combinations.

```
def run_experiment(speed1, speed2, duration):
    command = f"java -cp . Main Lab1.map {speed1} {speed2}"
    try:
        process = subprocess.Popen(command, shell=True,
stdout=subprocess.PIPE, stderr=subprocess.PIPE)
        stdout, stderr = process.communicate(timeout=duration)
        return_code = process.returncode
        stdout = stdout.decode('utf-8', errors='ignore')
        stderr = stderr.decode('utf-8', errors='ignore')
        if return_code != 0:
            print("Java program exited with error code:", return_code)
            print("Error message:")
            print(stderr)
    except subprocess.TimeoutExpired:
        process.kill()
    except Exception as e:
        print(e)

def main():
    speeds = range(25, 0, -1)
    results = []

    for speed1 in speeds:
        for speed2 in speeds:
            if speed1 > 10 and speed2 > 10:
                duration = 60
            else:
                duration = 180
            print("Testing with speeds: Train1={}, Train2=
{}".format(speed1, speed2))
            start_time = time.time()
            run_experiment(speed1, speed2, duration)
            end_time = time.time()

            print("Duration: {:.2f} seconds".format(end_time -
start_time))
            print("---")

if __name__ == "__main__":
    main()
```

To make it work, modifications are needed in the readLoop() method of TSimInterface.java. After reporting a trainEvent, the program should execute System.exit(1) to terminate.

```
...
trainVec.set(trainId, tEvent);
reportTrainEvent(tEvent);
System.exit(1);
...
```

And the debug mode should be turned off in Main.java to prevent the program from printing out the information about the sensor events.

```
...
TSimInterface.getInstance().setDebug(false);
...
```