Parking Manager

This project has been tested and compiled only with Java 8.

Compile parent project

If you what to use the Java API as a maven dependency please checkout the project is folder "parking" and run :

```
mvn install
```

Install to client using maven

The preferred way to start the project in client side is to add the dependency to your client project.

```
<!-- The client -->
<dependency>
        <groupId>fr.walidg.park</groupId>
        <artifactId>parking</artifactId>
        <version>1.0.0-SNAPSHOT</version>
        <scope>compile</scope>
</dependency>
```

Install to client using Java dependency (Alternative)

Without maven, please add to your project all files in "WithoutMaven" folder in client your build path.

Usage

In the client application we need to create a ParkingLotBuilder to initiate a parking.

```
                ParkingLotBuilder plb = new ParkingLotBuilder();
                ParkingLotManager plm = plb
                                .withElectric20Slots(1)
                                .withElectric50Slots(0)
                                .withPetrolSlots(10)

        .withPrincing(ParkingLotBuilder::FixedAndTimePricing)
                                .build();
```

Options

withElectric20Slots

Number of parking Slots available for Electric 20KW cars

- Type: `Integer`
- Default: `0`

withElectric50Slots

Number of parking Slots available for Electric 50KW cars

- Type: `Integer`
- Default: `0`

withPetrolSlots

Number of parking Slots available for petrol cars

- Type: `Integer`
- Default: `0`

withPrincing

Princing function, can be a static or not function

- Type: `Function<Slot, Float>`
- Default: `s -> 10.5f` Build in function can be used as :

```
ParkingLotBuilder::TimeOnlyPricing
ParkingLotBuilder::FixedAndTimePricing
```

Hypothesis

- No cheating user (ex. changing Vehicle plates inside the parking)
- Pricing function is "safe" (thread-safe and no runtime exception)
- No casting issue when converting price (duration of stay must be reasonable), same for number of slots.

I/O parking

- Create a Vehicle from import fr.walidg.park.parking.ParkingClient.*.
- Request a slot from the parking using plm.getASlot(vehicle).
- Free the slot to exit the parking using plm.freeASlot(vehicle).

```
Electic20KwVehicule v = new Electic20KwVehicule("ID");
Slot s = plm.getASlot(v);
Float price = plm.freeASlot(v);
System.out.println(String.format("The price is :  %10.2f €", price));
```

Pricing

The pricing function takes in argument a Slot instance occupied by the car and returns the expected price. The slot has a member "occupiedSince" that is the DateTime of the entering.

- For static method please have a look to this example

```
        public static Float FixedAndTimePricing(Slot s) {
                LocalDateTime now = LocalDateTime.now();
                long diff = Duration.between(s.getOccupiedSince(),
now).toHours();
                return (float) (Math.round((FIXED_ENTRY_PRICE + diff  *
PRICE_PER_HOUR) * 100.0) / 100.0);
        }
```

- For dynamic behavior :

```
import fr.walidg.park.parking.ParkingLotBuilder;
import fr.walidg.park.parking.ParkingLotManager;
import fr.walidg.park.parking.ParkingStructure.entity.Slot;

public class Test {
        public Test() {

        }
        public Float freepricing_price(Slot s) {
        //Here dynamic code
                return 0f;

        }
        public static void main(String[] args) {
                Test t = new Test();
                ParkingLotManager plm = new ParkingLotBuilder()

        .withPrincing(t::freepricing_price)

        .build();
        }
}
```