1. Boat Racing Simulator

This document defines the workshop for "Java OOP" course @ Software University. Please submit your solution (source code) of below described problem in Judge.

For this workshop, your task is to write a **boat racing simulator** application.

2. Overview

The Boat Racing Simulator holds information about **Boats, Boat engines** and a **Race**.

The system contains methods for **creating boat engines, creating boats, opening a race, signing up boats** (for the race)**, starting the race** and **ending the input.**

There are four types of boats:

- **Row Boat** which has a **model, weight** and **oars.**
- **Sail Boat** which has a **model, weight** and **sail efficiency.**
- **Power Boat** which has a **model, weight** and **two boat engines.**
- **Yacht** which has a **model**, **weight, boat engine** and **cargo weight.**

A boat's **model** is **unique** – there cannot be two boats with the same model.

A **Boat Engine** has **model** and **output,** there are two types of engines **Jet Engines** and **Sterndrive Engines**. An engine receives **horsepower** and **displacement** and calculates its own **output**. The formulas are as follows:

Engine Type	Output
Jet Engines	(Horsepower * 5) + Displacement
Sterndrive Engines	(Horsepower * 7) + Displacement

A boat engine's **model** is **unique** – there cannot be two boat engines with the same model.

A **Race** contains **distance, wind speed, ocean current speed**, a collection of participants (boats that have signed up for the race) and an **AllowMotorboats** property that signifies if motor boats (boats which have an engine) are allowed. A **Race** also contains methods for adding to and returning the collection of participants, adding participants should check if a participant with the same **Model** has already been registered for the race and throw a **DuplicateModelException** in such a case.

CreateBoatEngine tries to make a new boat engine of the specified type and if the parameters passed are valid saves the resulting engine in the system.
Note: The only valid engine types at the current time are **"Jet"** and **"Sterndrive"**.

There are four methods for creating boats - CreateRowBoat, CreateSailBoat, CreatePowerBoat and CreateYacht**,** each tries to create a new boat of its type and if the parameters passed are valid saves the resulting boat in the system.

OpenRace creates a race with the specified parameters and tries to set the current race to the created one. There can only be one race at a time, if there is an already set up race the command fails.

SignUpBoat attempts to sign the boat with the specified **model** into the current race, if the type of the boat does not meet the requirements of the race (i.e. the boat is a Yacht and the race does not allow **Motorboats**) the command fails and throws an exception.

StartRace starts the current race, all boats that signed up for the current race compete and the 3 with the fastest time for the race are printed as winners, after this command the current race is cleared. The way to calculate the speed (m/s) for the current race for each boat type is as follows:

Boat Type	Speed
Row Boat	(Oars * 100) - Boat Weight + Race Ocean Current Speed
Sail Boat	(Race Wind Speed * (Boat Sail Efficiency / 100)) – Boat's Weight + (Race Ocean Current Speed / 2)
Power Boat	(Engine 1 Output + Engine 2 Output) - Boat's Weight + (Race Ocean Current Speed / 5);
Yacht	Boat Engine Output - (Boat Weight + Cargo Weight) + (Race Ocean Current Speed / 2);

It is important to note that the resulting speed for boats **CAN be negative or 0,** in that situation the boat will **NEVER finish** and in place of its Time it should print **"Did not finish!". If two or more boats have the same time/did not finish, their placements are determined by the order of signing up to the race.** (check the sample output to gain a better idea of how it works).

End command ends the input stream.

System Design

The core of the system is the **engine**, it reads lines from the standard input (console) splits each into command name and parameters and passes them to a **Command Handler,** the engine also **catches any exceptions** and prints their message on the standart output(console).

A sample input line is shown below:

CommandName\value1\value2\..

Values will consist only of **Latin letters** and **numbers**. The **command name** and **values** will be seperated by a single **"\". All commands given will be correct** (will contain only correct command names, number of parameters and parameter types), **you don't have to check them specifically**.

The **Command Handler** delegates all actions to a **controller.** Using the parsed input from the engine, it calls actions from the **controller** and optionally **parses the passed parameters** if needed.

In order to work with model collections, the project has a **data layer**. The data layer consists of **repositories**. A repository contains objects of the same type and provides methods for the following:

- **Getting an item** by its unique Model (should throw **NonExistantModelException** if an item with the given model does not exist in the database).
- **Adding** a new item (should throw **DuplicateModelException** if an item with the same model already exists in the database).

A **database** class combines all repositories defined for the application.

The **controller** contains the main business logic of the application. It contains a **database** and all the **actions**. An **action** is a public method which either returns a **string result** or throws an **exception** and can optionally accept parameters.

The **controller** checks the validity of the current action. For example, if a command for starting a race is received while there is no currently set race the system will reject the request and throw a **NoSetRaceException** with the message **"There is currently no race set."**.

Models are classes containing information about the real-world objects the system works with. The system should support all the above mentioned types of **Boats, Boat Engines** and **Race**:

Not all models are valid. The validation rules for the models are given below:

- A Boat's model must be at least 5 symbols long.
- A Boat Engine's model must be at least 3 symbols long.

In case the validation fails the system throws an **ArgumentException** with the message:

- **"Model's name must be at least [min model's length] symbols long."**

- A Boat's Weight must be a positive (non-zero) integer.
- A Row Boat's Oars must be a positive (non-zero) integer.
- A Yacht's Cargo Weight must be a positive (non-zero) integer.
- A Boat Engine's Horsepower must be a positive (non-zero) integer.
- A Boat Engine's Displacement must be a positive (non-zero) integer.
- A Race's Distance must be a positive (non-zero) integer.

In case the validation fails the system throws an **ArgumentException** with the message:

- **"[Parameter's name] must be a positive integer."**

Where Parameter's name can only be one of the following **"Weight", "Oars", "Cargo Weight", "Horsepower", "Displacement", "Distance".**

A Sail Boat's Sail Effectiveness must be between [1…100]. In case the validation fails, the system throws an **ArgumentException** with the message:

- **"Sail Effectiveness must be between [1...100]."**

System Functionality

The boat racing simulator system contains the following commands:

- **`CreateBoatEngine\<model>\<horsePower>\<displacement>\<type>`**
  Creates a new boat engine of the specified type model, horsepower and displacement.

Case	Message	Exception
Success	Engine model [model] with [horsepower] HP and displacement [displacement] cm3 created successfully.	None

- **`CreateRowBoat\<model>\<weight>\<oars>`**
- **`CreateSailBoat\<model>\<weight>\<sailEfficiency>`**
- **`CreatePowerBoat\<model>\<weight>\<boatEngine>\<secondEngine>`**
- **`CreateYacht\<model>\<weight>\<boatEngine>\<cargoWeight>`**
  Depending on the method creates a new Row Boat,Sail Boat, Power Boat or Yacht with the given parameters.

Case	Message	Exception
Success	[Boat type] with model [model] registered successfully.	None

- **`OpenRace\<distance>\<windSpeed>\<oceanCurrentSpeed>\<allowsMotorboats>`**
  Creates a new Race with the specified parameters and tries to set it as the current Race, if the currentRace is already set, the command fails and throws and exception.

Case	Message	Exception
Success	A new race with distance [distance] meters, wind speed [windSpeed] and ocean current speed [oceanCurrentSpeed] has been set.	None
The current race has already been set.	The current race has already been set.	RaceAlreadyExistsException

- **`SignUpBoat\<model>`**
  Signs up the boat with the specified model in the current Race. If there is no currently set up Race or the type of boat is not allowed by the race, the command fails and throws an exception.

Case	Message	Exception
Success	Boat with model [model] has signed up for the current Race.	None
The current Race has not been set.	There is currently no race set.	NoSetRaceException
The specified boat does not meet the race constraints.	The specified boat does not meet the race constraints.	ArgumentException

- **StartRace**
  Start the current Race, each participant's time for completing the race is calculated and the 3 with the best times (smallest times) are printed in ascending order. The current Race should be cleared(removed) after this command.
  Note: Time should be **rounded to exactly two decimal places**.

Case	Message	Exception
Success	First place: [typeOfBoat] Model: [model] Time: [boatsRaceTime] Second place: [typeOfBoat] Model: [model] Time: [boatsRaceTime] Third place: [typeOfBoat] Model: [model] Time: [boatsRaceTime]	None
The current Race has not been set.	There is currently no race set.	NoSetRaceException
There are less than 3 boats registered for the race.	Not enough contestants for the race.	InsufficientContestantsException

Model the system and all entities using the best established practices in object-oriented design and object-oriented programming.

The input should be read from the console. The output is written to the console. The input and output formats have been specified above.

3. Sample Input 1

```
CreateBoatEngine\GPH01\250\100\Jet
CreateBoatEngine\GPH02\150\150\Sterndrive
CreateRowBoat\Rower15\450\6
CreatePowerBoat\PB150\2200\GPH01\GPH02
CreateSailBoat\SailBoatPro\200\98
OpenRace\1000\10\5\true
SignUpBoat\SailBoatPro
SignUpBoat\Rower15
SignUpBoat\PB150
StartRace
End
```

4. Sample Output 1

```
Engine model GPH01 with 250 HP and displacement 100 cm3 created successfully.
Engine model GPH02 with 150 HP and displacement 150 cm3 created successfully.
Row boat with model Rower15 registered successfully.
Power boat with model PB150 registered successfully.
Sail boat with model SailBoatPro registered successfully.
A new race with distance 1000 meters, wind speed 10 m/s and ocean current speed 5 m/s has been set.
Boat with model SailBoatPro has signed up for the current Race.
Boat with model Rower15 has signed up for the current Race.
Boat with model PB150 has signed up for the current Race.
First place: PowerBoat Model: PB150 Time: 2.85 sec
Second place: RowBoat Model: Rower15 Time: 6.45 sec
Third place: SailBoat Model: SailBoatPro Time: Did not finish!
```

5. Sample Input 2

```
CreateBoatEngine\SI20\200\100\Sterndrive
CreateBoatEngine\SI10\300\200\Jet
CreateRowBoat\MasterRower10\200\4
CreateRowBoat\MasterRower12\100\0
StartRace
```

```
CreateRowBoat\MasterRower11\200\4
CreatePowerBoat\Turbo220\1550\SI20\SI10
CreateYacht\Luxury101\1000\SI20\150
SignUpBoat\MasterRower11
CreateSailBoat\SailBoatPro\80\98
OpenRace\1500\150\10\false
SignUpBoat\SailBoatPro
SignUpBoat\Turbo220
OpenRace\2000\80\80\false
StartRace
SignUpBoat\MasterRower11
SignUpBoat\MasterRower10
SignUpBoat\Luxury101
StartRace
End
```

6. Sample Output 2

```
Engine model SI20 with 200 HP and displacement 100 cm3 created successfully.
Engine model SI10 with 300 HP and displacement 200 cm3 created successfully.
Row boat with model MasterRower10 registered successfully.
Oars must be a positive integer.
There is currently no race set.
Row boat with model MasterRower11 registered successfully.
Power boat with model Turbo220 registered successfully.
Yacht with model Luxury101 registered successfully.
There is currently no race set.
Sail boat with model SailBoatPro registered successfully.
A new race with distance 1500 meters, wind speed 150 m/s and ocean current speed 10 m/s has been set.
Boat with model SailBoatPro has signed up for the current Race.
The specified boat does not meet the race constraints.
The current race has already been set.
Not enough contestants for the race.
Boat with model MasterRower11 has signed up for the current Race.
Boat with model MasterRower10 has signed up for the current Race.
The specified boat does not meet the race constraints.
First place: RowBoat Model: MasterRower11 Time: 7.14 sec
Second place: RowBoat Model: MasterRower10 Time: 7.14 sec
```

Third place: SailBoat Model: SailBoatPro Time: 20.83 sec

7. Bonus: Implement a GetStatistic Command

Implement a **GetStatistic** command which prints the percentage of participants for each boat type in the current race **sorted in alphabetical order** and **rounded to two decimal places**. A third zero test is provided speciffically for this command. Check the example bellow to get a better understanding of the task.

Example Input

CreateBoatEngine\Engine1\100\100\Sterndrive

CreateBoatEngine\Engine2\150\100\Jet

CreateSailBoat\ExampleSailBoat\50\90

CreateRowBoat\ExampleRowBoat\100\4

CreateRowBoat\ExampleRowBoat2\120\6

CreatePowerBoat\ExamplePowerBoat\800\Engine1\Engine2

CreateYacht\ExampleYacht\700\Engine2\150

OpenRace\100\10\5\true

SignUpBoat\ExampleSailBoat

SignUpBoat\ExampleRowBoat

SignUpBoat\ExampleRowBoat2

SignUpBoat\ExamplePowerBoat

SignUpBoat\ExampleYacht

GetStatistic

StartRace

End

Example Output

Engine model Engine1 with 100 HP and displacement 100 cm3 created successfully.

Engine model Engine2 with 150 HP and displacement 100 cm3 created successfully.

Sail boat with model ExampleSailBoat registered successfully.

Row boat with model ExampleRowBoat registered successfully.

Row boat with model ExampleRowBoat2 registered successfully.

Power boat with model ExamplePowerBoat registered successfully.

Yacht with model ExampleYacht registered successfully.

A new race with distance 100 meters, wind speed 10 m/s and ocean current speed 5 m/s has been set.

Boat with model ExampleSailBoat has signed up for the current Race.

Boat with model ExampleRowBoat has signed up for the current Race.

Boat with model ExampleRowBoat2 has signed up for the current Race.

Boat with model ExamplePowerBoat has signed up for the current Race.

Boat with model ExampleYacht has signed up for the current Race.

PowerBoat -> 20.00%

RowBoat -> 40.00%

SailBoat -> 20.00%

Yacht -> 20.00%

First place: PowerBoat Model: ExamplePowerBoat Time: 0.12 sec

Second place: RowBoat Model: ExampleRowBoat2 Time: 0.21 sec

Third place: RowBoat Model: ExampleRowBoat Time: 0.33 sec