Lab 12 Object-Oriented Programming 2

Note: The underlined text is an input data.

1. **Create class `Bus` stored in file `Bus.php` that extends class `Car` from full example of lecture.** Class `Bus` can load the passengers into it by using method `load($number_of_passenger)` and unload the passengers by using method `unload($number_of_passenger)` then the fuel used can be calculated per passenger run from the following formula. Moreover, each `Bus` has its own capacity (maximum number of passengers) setting by constructor.

$$fuel\ used\ per\ passenger\ run\ (L) = \frac{passenger\ distance\ (km)}{120} \times \frac{piston\ volume\ (cc)}{1000} + \frac{70 \times number\ of\ passengers \times passenger\ distance\ (km)}{10000}$$

Remark: Beware, the number of passengers can be changed over time so you must have property to keep fuel used, calculating fuel used on showing is not correct.

The method `showLongInfo()` will print out the following format, same as `Car` but extends with number of current passenger.

```
Owner: owner
Running distance:    dist km
Fuel used:           fuel L
Current passenger:   number_of_passengers
```

Then write the program that waits for inputting owner name, piston volume and capacity then wait for the following command.

```
0 stop engine
1 start engine
r run for the given km
+ load the given number of passengers into bus
- unload the given number of passengers out of bus
i show information (engine is off only)
e exit
h print this help
```

If the number of passengers greater than capacity it will print out the following message to STDERR then return false, in method `load()`.

```
Number of passengers greater than capacity!!!
```

If the number of passengers less than `0` it will print out the following message to STDERR then return false, in method un`load()`.

```
Number of passengers less than 0!!!
```

Example 01: `php ass-01.php`

```
Input (owner cc capacity): Stieve 30000 40
command (h for help): h
 0 stop engine
 1 start engine
 r run for the given km
 + load the given number of passengers into bus
 - unload the given number of passengers out of bus
 i show information (engine is off only)
 e exit
 h print this help
command (h for help): r 50
Cannot run, engine is off!!!
command (h for help): 1
command (h for help): r 200
command (h for help): - 20
Number of passengers less than 0!!!
command (h for help): + 30
command (h for help): r 1000
command (h for help): + 20
Number of passengers greater than capacity!!!
command (h for help): r 500
command (h for help): - 10
command (h for help): r 500
command (h for help): + 20
command (h for help): r 1000
command (h for help): - 30
command (h for help): i
Cannot show, engine is on!!!
command (h for help): 0
command (h for help): i
Owner: Stieve
Running distance:          3,200 km
Fuel used:              1,465.00 L
Current passengers:           10
command (h for help): e
```

2. Create class **`Pet`** stored in file **`Pet.php`** that implements interface **`Runnable`** and **`ShowInfo`** and class **`PetLover`** stored in file **`PetLover.php`** that extends class **`Person`** and implements interface **`ShowInfo`** from full example of lecture.

   Class `Pet` consist of the following members.

   - `$name`: assigned by constructor.
   - `$distance`: accumulated distance from method `runFor()`.
   - `getName()`: get the pet name.
   - `showInfo()`: print out the following format.

   ```
   Name:                 pet_name
   ```

   - `showLongInfo()`: print out the following format.

   ```
   Name:                 pet_name
   Running distance:     dist km
   ```

   Class `PetLover` consist of the following members (extends from parentclass).

   - `takePet(Pet $pet)`: take an instance of `Pet` with his/her.
   - `releasePet(Pet $pet)`: will release the given instance of `Pet` from taken pets.
   - `showInfo()`: print out the following format, same as .

   ```
   Name:                 pet_lover_name
   ```

   - `showLongInfo()`: print out the following format.

   ```
   Name:                 pet_lover_name
   Running distance:     dist km
   Current taken pets: pet1_name, pet2_name, ...
   ```

   Remark: `PetLover` can take more than one pet on the same time.

   If `PetLover` do `runFor()` all taken pets will `runFor()` together so you must override method `runFor()` in class `PetLover`.

   All pets always have the difference name.

   You can use === to check for the same object.

Write the program that reads the input file with the following format.

```
pet_lover_name
number_of_pets
pet1_name
pet2_name
...
number_of_command
command1 value1
command2 value2
...
```

The commands consist of:

```
t  take the given pet name
re release the given pat name
r  pet lover run for the given km
```

Example Input: `ass-02-input.txt`

```
Susan
3
Red
Black
White
13
r 3
t Red
r 2
t Black
t Red
r 5
re Red
re White
r 2
t White
t Red
r 3
re Black
```

Beware that command can take pet that already taken or can release non-taken pet and those command is no effect. After all commands have been executed then show information from `showLongInfo()` for pet lover and all pets.

Example 01: `php ass-02.php ass-02-input.txt`

```
Name: Susan
Running distance:            15 km
Current taken pets: White, Red
========================================
Name: Red
Running distance:            10 km
----------------------------------------
Name: Black
Running distance:            10 km
----------------------------------------
Name: White
Running distance:             3 km
----------------------------------------
```