COS126 Princeton University Programming Exam 2 Spring '23

Before the exam. Read this page of instructions before the exam begins. Do not start the exam (or read the next page) until instructed to do so.

Duration. Once the exam begins, you have 80 minutes to complete it. You may not submit after time has been called.

Submission. Submit your solutions on TigerFile using the link from the Exams page. You may submit multiple times (but only the last version will be graded).

Check Submitted Files. You may click the ***Check Submitted Files*** button to receive *partial feedback* on your submission. We will attempt to provide this feature during the exam, but you should not rely upon it.

Grading. Your program will be graded primarily on correctness. You should comment the code as well. Efficiency and clarity will also be considered. You will receive partial credit for a program that implements some of the required functionality. You will receive a substantial penalty for a program that does not compile.

Allowed resources. During the exam you may use only the following resources: course textbook, companion booksite, course website, course Ed, precepts, your course notes, and your code from the programming assignments. For example, you may not use StackOverflow, Google, ChatGPT, etc.

No collaboration or communication. Collaboration and communication during this exam are prohibited, except with course staff. A staff member will be outside the exam room to answer clarification questions.

No electronic devices or software. Software and computational/communication devices are prohibited, except to the extent needed for taking this exam (such as a laptop, browser, and IntelliJ). For example, you must close all unnecessary applications and browser tabs; disable notifications; and turn off your cell phone.

Honor Code pledge. Write and sign the Honor Code pledge by typing the text below in the file **`acknowledgments.txt`**.

> *I pledge my honor that I will not violate the Honor Code during this examination.*
>
> Electronically sign it by typing **`/s/`** followed by your name.

After the exam. Discussing or communicating the contents of this exam before solutions have been posted is a violation of the Honor Code.

Background. A coin-operated parking meter is a type of device commonly used to control and monitor parking time for cars. When coins are inserted, time is added to the parking meter based on the *rate.* For example, adding 50 US cents to a 25 cent per hour parking meter adds two (2) hours to the meter. A timer ticks down minute by minute until it reaches zero (0) minutes. More coins can be inserted to increase the time on a given parking meter. For this exercise, all parking meters have a maximum of five (5) hours.

Problem. Write a mutable data type `ParkingMeter.java` that represents a US coin-operated parking meter. Each parking meter is configured with a rate in cents/hour, e.g., 10 cents/hour or 40 cents/hour, etc. Rates can range between 1 cent and 99 cents, inclusive, per hour. A parking meter is also configured with an initial time remaining - ranging between 0 and 300 minutes, inclusive. Additional time can be added to a parking meter by inserting more coins, although the time can never exceed five (5) hours (i.e., 300 minutes).

Implement the following API. (Suggestions: implement the constructor and instance methods in the order in which they appear; test all methods in the **`main()`** as you go.) You must not modify the (public) API.

`public class ParkingMeter`	
`public ParkingMeter(int rate, int initial)`	*Creates a meter with rate cents/hour and the initial remaining time in minutes*
`public String toString()`	*Returns a string representation of this parking meter*
`public int tic()`	*Decrements the parking meter by one (1) minute and returns the remaining minutes.*
`public boolean lessTime(ParkingMeter that)`	*Returns true if this parking meter has less time remaining than that parking meter*
`public int insert(int cents)`	*Adds time to the parking meter, according to the given cents and the rate; returns the number of minutes actually added.*
`public static void main(String[] args)`	*Tests all instance methods in this class*

We provide a template **`ParkingMeter.java`** that you can modify. This template contains the API and compiles. Ensure that your code **compiles successfully** before submitting. Also, you should **resolve all Checkstyle messages**.

The following page provides more information about the required behavior.

- **The two-argument constructor.** Throws an **`IllegalArgumentException`** if either integer argument is outside its bounds (**`rate`** must be between 1 and 99, inclusive and **`time`** must be between 0 and 300, inclusive).

- **String representation.** The format is:

  ```
          HH:MM $.CC
  ```

  where **`HH`** is the hours remaining (2 digits), followed by a colon (**`:`**), **`MM`** is the minutes remaining (2 digits), followed by a space, dollar sign and decimal point (**`$.`**), **`CC`** is the rate, i.e., cents per hour (2 digits). Examples:

  - **`01:00 $.10`**
  - **`00:00 $.01`**
  - **`04:22 $.99`**

  Hint: Use the `String.format` method (which is similar to `StdOut.printf`). For example:

  ```
      String.format("%02d:%02d $.%02d", 1, 1, 1)
  ```

  returns the `String` value:

  ```
      "01:01 $.01"
  ```

- **Tic.** Decrement one minute from the remaining time. The remaining time must never be negative.

- **Inserting coins.** Inserting coins into a parking meter increases the amount of time based on the configured parking meter rate. Time is always an integer number of minutes, not a fractional. For example, suppose the rate for a parking meter is 10 cents/hour:
  - Inserting 20 cents increases the time by 120 minutes:
    ```
        Math.floor(20.0 / 10.0 * 60) = 120
    ```
  - Inserting 15 cents increases the time by 90 minutes:
    ```
        Math.floor(15.0 / 10.0 * 60) = 90
    ```
  - Inserting 25 cents increases the time by 150 minutes:
    ```
        Math.floor(25.0 / 10.0 * 60) = 150
    ```
  - Inserting 11 cents increases the time by 66 minutes:
    ```
        Math.floor(11.0 / 10.0 * 60) = 66
    ```

  The total time must never exceed five (5) hours. For example, if the rate is 10 cents/hour:

  - If the remaining time is 120 minutes, inserting 10 cents increases the remaining time by 60 minutes (to 180 minutes) and returns 60 minutes
  - If the remaining time is 292 minutes, inserting 10 cents increases the remaining time by only 8 minutes (to 300 minutes) and returns 8 minutes.

  Also, throw an **`IllegalArgumentException`** if the given value for cents is **≤ 0**.

- **Test client.** The **`main()`** method must call each constructor and instance method directly for testing (e.g., by printing results to standard output).