



&lt; Anterior



Siguiente &gt;

## HW06

Marcar esta página

### Las tareas calificadas están bloqueadas

Cámbiate a la opción verificada para obtener acceso a funciones bloqueadas como esta y aprovechar al máximo tu curso.



Cuando te cambias a la opción verificada, tú:

- ✓ Obtén un **certificado verificado** de finalización para compartirlo en tu currículum
- ✓ Desbloquea el acceso a todas las actividades del curso, incluidas las **tareas calificadas**
- ✓ **Acceso completo** al contenido y los materiales del curso
- ✓ Desbloquea el acceso a todas las actividades del curso, incluidas las **tareas calificadas**
- ✓ **Acceso completo** al contenido y los materiales del curso, incluso después de que finalice el curso
- ✓ Apoye a nuestro **misión** en edX

**Opción verificada \$199**Homework fecha límite Jul 18, 2023 08:10 -05 Vencidos

### Problem Description

Hello! Please make sure to read all parts of this document carefully.

In this assignment, you will be applying your knowledge of abstract classes, inheritance, polymorphism, file I/O and exceptions. For this homework, you will be simulating a veterinary clinic. You will create a **Pet.java**, **Dog.java**, **Cat.java**, **InvalidPetException.java**, and a **Clinic.java** file; the **Clinic.java** file will treat and keep a record of the Pet patients it receives!

### Solution Description

Create files **Pet.java**, **Dog.java**, **Cat.java**, **InvalidPetException.java** and **Clinic.java**. Each file will have instance fields, methods, and constructors.

## Pet.java

This class represents any pet that would seek consultation from the clinic.

### Variables:

- `String name`
- `double health`
  - A percentage value ranging from 0.0 to 1.0
- `int painLevel`
  - Ranges from 1 to 10

### Constructor:

- `Pet(String name, double health, int painLevel)`
  - `health`
    - If `health` passed in is greater than 1.0, set `health` to 1.0
    - If `health` passed in is less than 0.0, set `health` to 0.0
  - `painLevel`
    - If `painLevel` passed in is greater than 10, set `pain level` to 10
    - If `painLevel` passed in is less than 1, set `pain level` to 1

### Methods:

- getters for all instance fields, which should be camelCase with the variable name, e.g. a variable named `hello` should have a getter `getHello()`
- `int treat():`
  - Should be an **abstract** method that returns the time taken (in minutes) to treat the pet
- `void speak():`
  - This method prints "Hello! My name is " with the pet's name
  - If `painLevel` is greater than 5 prints the message in UPPERCASE
- `boolean equals(Object o):`
  - Two Pet objects are equal if their names are the same
  - Note: You can assume you will not encounter two pets with the same name
- `heal():`
  - Should be **protected** to prevent access by external classes
  - Sets `health` to 1.0
  - Sets `painLevel` to 1

## Dog.java

Since a Dog is also a Pet, this class must inherit from parent class Pet. This class is concrete.

**Variables:**

- double droolRate

**Constructors:**

- Dog(String name, double health, int painLevel, double droolRate)
  - droolRate - If droolRate is less than or equal to zero, set drool rate to 0.5
- Dog(String name, double health, int painLevel)
  - Default droolRate is 5.0

**Methods:**

- getters for all instance fields, which should be camelCase with the variable name, e.g. a variable named hello should have a getter getHello()
- **int treat():**
  - Should heal()
  - Returns the time taken (in minutes) to treat the pet. Round values up.
    - if droolRate is less than 3.5, the minutes for treatment is  $(\text{painLevel} * 2) / \text{health}$
    - if droolRate is in between 3.5 and 7.5 inclusive, the minutes for treatment is  $\text{painLevel} / \text{health}$
    - if droolRate is greater than 7.5, the minutes for treatment is  $\text{painLevel} / (\text{health} * 2)$
- **void speak():**
  - Calls parent method
  - Prints "bark" number of times of the painLevel
    - e.g.: if painLevel = 3
      - Prints "bark bark bark"
    - ALL UPPERCASE if painLevel is greater than 5, not inclusive
- **boolean equals(Object o):**
  - Uses the equals() method in Pet as part of the decision-making with the additional condition of droolRate being the same

**Cat.java**

Since a Cat is also a Pet, this class must inherit from parent class Pet. This class is concrete.

**Variables:**

- int miceCaught

**Constructors:**

- Cat(String name, double health, int painLevel, int miceCaught)
  - miceCaught

- If miceCaught passed in is less than 0, set miceCaught to 0
  - `Cat(String name, double health, int painLevel)`
    - Default miceCaught is 0
- Methods:**
- getters for all instance fields, which should be camelCase with the variable name, e.g. a variable named hello should have a getter `getHello()`
  - `int treat():`
    - Should heal()
    - Returns the time taken (in minutes) to treat the pet. Round all values up.
      - if number of miceCaught is less than 4, the minutes for treatment is equal to  $(\text{painLevel} * 2)/\text{health}$
      - if miceCaught is in between 4 and 7 inclusive the minutes for treatment equals  $\text{painLevel}/\text{health}$
      - if miceCaught is greater than 7, the minutes for treatment equals  $\text{painLevel}/(\text{health} * 2)$
  - `void speak():`
    - Calls parent method
    - Prints "meow" number of times of miceCaught
      - Eg: if miceCaught = 3
        - Print "meow meow meow"
      - ALL UPPERCASE if painLevel is greater than 5, not inclusive
  - `boolean equals(Object o):`
    - Uses the `equals()` method in Pet as part of the decision-making with the additional condition of miceCaught being the same

### InvalidPetException.java

An **unchecked exception** with two constructors

#### Constructors

- `InvalidPetException()` has message "Your pet is invalid!"
- `InvalidPetException(String s)` has message s

### Clinic.java

This is a class representing the vet clinic.

#### Variables

- `File patientFile`
  - File with patient information
- `int day`

#### Constructors

- `Clinic(File file)`
  - File that contains patient info - assign to `patientFile`
  - Name
  - Type of pet (includes pet info)
  - Appointment Info
    - `timeIn(military time)`
    - `health(before Treatment)`
    - `painLevel(before Treatment)`
    - `TimeOut(military time)`
  - `TimeOut(military time)`
  - Day initialized to 1
- `Clinic(String fileName)`
  - String includes filename extension – don't add ".csv"
  - Chains to the other constructor

#### Methods

- `String nextDay(File f) throws FileNotFoundException`  
`String nextDay(String fileName) throws FileNotFoundException`
  - Reads File f that contains the name, type of pet, and time of the appointments for the day
    - See example file "Appointments.csv" for the format
    - Eg: If there was a Cat Chloe, with a miceCaught count of 5, scheduled for 2:30 pm, Chloe's information in Appointments.csv would look like:  
Chloe,Cat,5,1430
    - You will have one file for each different day
  - Use a Scanner object to take in user input
  - Print "Consultation for [name] the [typeOfPet] at [time].\nWhat is the health of [name]?\\n"
  - If `typeOfPet` is not valid (i.e. not a Dog or Cat, case-sensitive) throw `InvalidPetException`
    - Do not catch the exception in your code! The caller of the method should handle the exception.
  - Take in user input for health
    - If input is not a number, continue prompting user until they provide a number
  - Print "On a scale of 1 to 10, how much pain is [name] in right now?\\n"
  - Take in user input for `painLevel`
    - If input is not a number, continue prompting user until they provide a number
  - Call `speak()`

- treat pet
- Calculate time out (there exists a method for this)
- Note: Don't try to read the file and write to it at the same time – this method is intended only to read the file.
- Don't forget the increment the day!
- Returns a String with patient information to be used when treating patients and updating the file.
- The string being returned should hold the updated information for all patients seen in the day separated by a newline character.
- Each appointment should be formatted as follows:
  - [Name],[Species],[DroolRate/MiceCaught],[Day],[EntryTime],[ExitTime],[InitialHealth],[InitialPainLevel]
  - E.x.: If there are 2 appointments on day 2:
    - Appointment 1 on Day 2:
      - Dog Dobie with droolRate 2.7
      - Entry time: 1715 (5:15 pm) and Exit time: 1735 (5:35 pm)
      - Health was 0.5 and painLevel was 5 before treating
    - Appointment 2 on Day 2:
      - Cat Marlin with miceCaught 84
      - Entry time: 1655 (4:55 pm) and Exit time: 1700 (5:00 pm)
      - Health was 0.4 and painLevel was 4 before treating

The output of nextDay would be:

```
Dobie,Dog,2.7,Day 2,1715,1735,0.5,5
Marlin,Cat,84,Day 2,1655, 1700,0.4,4
```

- boolean addToFile(String patientInfo)
  - Consumes a string representing a single appointment
  - Eg. In format:  
[Name],[Species],[DroolRate/MiceCaught],[Day],[EntryTime],[ExitTime],[InitialHealth],[InitialPainLevel]
  - Write info to patientFile
    - If old patient, only the **appointment info** should be added to the patient file, which includes:
      - Day #
      - Time in and time out
      - Health and pain
    - If new patient, all info should be added to the clinic's patient file
    - Assume the vet will never see two different pets with the same name
    - See Patients.csv for an example

- Returns true if the appointment info was successfully written, and false if an error occurs or a checked exception is caught
  - Note (cont'd): Don't try to read the file and write to it at the same time – this method is intended to rewrite the file.
- `String addTime(String timeIn, int treatmentTime)`
    - This method should only be accessible in the Clinic class
    - This method should calculate the time the patient's appointment ends
    - Return timeOut
    - Remember: timeIn and timeOut should be represented in military time
    - You can assume that timeIn and timeOut will **NOT** go across multiple days (ex. timeIn = "23:30" and timeOut = "00:30")

### **Example Output**

User input is **bolded**

Example output for this entry: Chloe,Cat,5,1430

Consultation for Chloe the Cat at 1430.

What is the health of Chloe?

**0.6**

On a scale of 1 to 10, how much pain is Chloe in right now?

**Six**

Please enter a number

On a scale of 1 to 10, how much pain is Chloe in right now?

**6**

HELLO! MY NAME IS CHLOE

MEOW MEOW MEOW MEOW MEOW

Reuse your code when possible. Certain methods can be reused using certain keywords.

### **Allowed Imports**

To prevent trivialization of the assignment, you are only allowed to import the following classes or packages.

`java.util.Scanner;`

java.io.File;

```
java.io.File,  
java.io.FileNotFoundException;  
java.io.IOException;  
java.io.PrintWriter;
```

## Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

- var (the reserved keyword)
- System.exit

---

## Grading

Homeworks are graded in an "all or nothing" manner. If your code is correct, you receive a 100 for the assignment; if it isn't, you receive a 0.

## Allowed Collaboration

When completing homeworks for CS1331 you may talk with other students about:

- What general strategies or algorithms you used to solve problems in the homeworks
- Parts of the homework you are unsure of and need more explanation
- Online resources that helped you find a solution
- Key course concepts and Java language features used in your solution

You may **not** discuss, show, or share by other means the specifics of your code, including screenshots, file sharing, or showing someone else the code on your computer, or use code shared by others.

---

## The Vocareum (code editor) interface has six main components:

- The **Drop-Down** in the top left. This lets you choose from multiple available files. Note that this drop-down will only be visible in assignments that require multiple files.
- The **Build / Run** button. For all assignments in this course, the build and run button will perform the same action: compile your code and run a file scan. Building and running your code will not count towards your total allowed submission attempts, therefore you are free to build / run as many times as needed.
- The **Submit** button. This will compile your code, run a file scan, grade your assignment, and output results to console. Note that for most assignments in this class, you will only be allowed a limited number of submissions. A submission is counted when the submit button is clicked, regardless of whether or not your code is able to compile or if there are any file issues. Therefore, **we highly recommend that you build or run your code before submitting to ensure that there are no issues that will prevent your code from being graded and that every submission attempt will generate meaningful results.**
- The **Reset** button. This will revert all your changes and reset your code to the default code template.
- The **Code Window**. This is where you will write your code. Again, We highly recommend copying the starter code and working in your preferred IDE.

- The **Output Window**. This window will appear whenever you build, run, or submit your code and will display the results for you to view.

### Vocareum Troubleshooting

We acknowledge that the Vocareum integration has some issues when submitting programs with multiple files. That is, the Vocareum interface may hide both the current file name and the combo box you need to select a file to edit. Unfortunately, this is beyond our control. Recall that the instructor recommends that you code and debug on your local JVM and submit in Vocareum mainly for grading. However, if you do need to edit in Vocareum, we have found the following workaround that you might try if you experience the stated problem.

1. Close any other homework from this course or another that uses the Vocareum integration.
2. Reload the webpage.
3. Open [Vocareum](#) in a new window. Keep the edX homework open: Vocareum won't visualize the homework description with the same formatting.
4. Select the course CS1331 and follow the instructions to start or continue working on the homework.
5. Look for where these four checkboxes appear: Files, README, Terminal, Source. Then, click the checkbox "Files." The README checkbox should uncheck automatically; if it doesn't, uncheck it.
6. You can now edit the files in the folder "work." Do not edit any files the extension "class" in the folder. Do not edit any file in any other folder or subfolder (such as "resource," "Submissions," or "workspace"). Do not create any additional files or folders.

**For additional help, please visit the Vocareum information page located in the course information module!**

---

For learners unable to access the Vocareum environment, this is the provided ClinicDriver.java, Patients.csv, and Appointments.csv file:

[ClinicDriver.java](#)

[Patients.csv](#)

[Appointments.csv](#)

◀ Anterior

Siguiente ▶

© Todos los Derechos están Reservados



**edX**

Acerca de  
Afiliados  
edX para negocios  
Open edX

**Legal**

Condiciones de Servicio y Código de Honor  
Política de privacidad  
Políticas de Accesibilidad

**Contáctanos**

Blog  
Contáctenos  
Centro de Ayuda  
Seguridad

[Carreras](#)  
[Noticias](#)

[Política de marcas](#)  
[Mapa del Sitio](#)  
[Política de cookies](#)  
[Your Privacy Choices](#)

[Kit Multimedia](#)



© 2023 edX LLC. All rights reserved.  
深圳市恒宇博科技有限公司 [粤ICP备17044299号-2](#)