# Упражнения: SOLID

## Информация за прогрес на поток

Рефакторирайте кода за тази задача, така че **Stream Progress Info** да може да работи с **различни видове** потоци. Първо се уверете, че работи с **Music**. Рефакторирайте кода, така че в бъдеще да **импортва един нов клас** с **BytesSent** и **Length** getter-и, когато се представи **нов вид поток**.

## Графичен редактор

Рефакторирайте кода за тази задача, така че **Graphic Editor да може да рисува всякакви видове фигури** без да проверява **какъв е видът на конкретната фигура**. В бъдеще ще бъдат добавяни нови фигури към системата, така че я пригответе за този момент. Когато се **добави нова фигура**, трябва да **добавите нов клас**.

## Принтер на детайли

Рефакторирайте кода за тази задача, така че **Detail Printer** да не се нуждае да пита **какъв тип служител е подаден**. Detail Printer трябва да отпечатва детайли за **всички видове служители**. Когато е добавен **нов вид служител**, трябва да **добавите нов клас.**

## Презареждане

Дадена е библиотека със следните класове

* Worker имплементира ISleeper
* Employee наследява Worker
* Robot наследява Worker
* RechargeStation

Ако инспектирате кода, ще видите, че някои от класовете имат методи, които **не могат да използват** (хвърлят UnsupportedOpperationException), което е ясна индикация, че кодът трябва да бъде **рефакториран**.

Рефакторирайте структурата, така че да спазите **принципа за разделяне на интерфейсите**.

### Насоки

Направете така че Robot да разширява Worker и в същото време да имплементира Rechargeable.

Text

Description automatically generated with medium confidence

## Logger

Напишете logging библиотека за **отпечатване на съобщения**. Интерфейсът за крайния потребител трябва да е следният:

|  |  |
| --- | --- |
| **Примерен код** | **Изход** |
| ILayout simpleLayout = new SimpleLayout();  IAppender consoleAppender =  new ConsoleAppender(simpleLayout);  ILogger logger = new Logger(consoleAppender);  logger.Error("3/26/2015 2:08:11 PM", "Error parsing JSON.");  logger.Info("3/26/2015 2:08:11 PM", "User Pesho successfully registered."); | 3/26/2015 2:08:11 PM - Info - User Peter successfully registered.  3/26/2015 2:08:11 PM - Error - Error parsing JSON. |

Logger отпечатва датата и часа (string) и съобщението (string).

### Архитектура на библиотеката

The library should have the following components:

Библиотеката трябва да има следните компоненти:

* **Layouts** – дефинират формата, в който съобщенията трябва да се добавят (напр. **SimpleLayout** отпечатва във формата "**<date-time> - <report level> - <message>**")
* **Appenders –** отговорни за добавянето на съобщенията някъде (напр. **Console**, **File**, т.н.)
* **Loggers** – съдържат методи за отпечатване от различен вид (предупреждения, грешки, информация, т.н.)

Когато **logger**-ът трябва да отпечата нещо, той **извиква** всички **appenders** и им "казва" да добавят съобщението. От своя страна, те добавят съобщението към конзолата или файла спрямо зададения **layout**.

### Изисквания

Библиотеката трябва да следва всички **SOLID** принципи:

* **Single Responsibility Principle (принцип за единствена отговорност)** – никой клас или метод не трябва да има повече от една отговорност
* **Open-Closed Principle (принцип** **отворен/затворен)** – библиотеката трябва да е **отворена за разширяване** (т.е. потребителят трябва да може да добавя собствени layouts/appenders/loggers), но **затворена** за пряка модификация
* **Liskov Substitution Principle (принцип на заместване на Лисков) -** детските класове не трябва да пречат на поведението на техния родител
* **Interface Segregation Principle (принцип за разделяне на интерфейсите)** - библиотеката трябва да използва **прости интерфейси**, които клиентът да имплементира
* **Dependency Inversion (принцип на обръщане на зависимостите)** – никой клас или метод не трябва да зависи директно от други модули (само от абстракции)

Избягвайте повторение на код. Задавайте смислени имена.

### Имплементации

The library should provide the following ready classes for the client:

Библиотеката трябва да осигурява следните **готови класове** за клиента:

* **SimpleLayout** – дефинира формата "**<date-time> - <report level> - <message>**"
* **ConsoleAppender** – добавя към конзолата, използвайки зададения **layout**
* **FileAppender** – добавя лог-а към файл, използвайки зададения **layout**
* **LogFile** – файлов клас, който логва съобщенията на string builder, използвайки метода **Write()**. Трябва да има **getter** за **размера**, който е **сумата** от **ASCII кодовете** на всички **символи от азбуката**, които съдържа.
* **Logger** – logger клас, който се използва за **отпечатване на съобщения**. Извиква всеки **appender**, когато трябва нещо да бъде отпечатано.

|  |  |
| --- | --- |
| **Примерен код** | **Изход** |
| var simpleLayout = new SimpleLayout();  var consoleAppender = new ConsoleAppender(simpleLayout);  var file = new LogFile();  var fileAppender = new FileAppender(simpleLayout, file);  var logger = new Logger(consoleAppender, fileAppender);  logger.Error("3/26/2015 2:08:11 PM", "Error parsing JSON.");  logger.Info("3/26/2015 2:08:11 PM", "User Pesho successfully registered."); | <log>  <date>3/31/2015 5:23:54 PM</date>  <level>Fatal</level>  <message>mscorlib.dll does not respond</message>  </log>  <log>  <date>3/31/2015 5:23:54 PM</date>  <level>Critical</level>  <message>No connection string found in App.config</message>  </log> |

Горният код трябва да отпечатва съобщенията и на **конзолата**, и във файла **log.txt** във формата от **SimpleLayout**.

### LogFile

Всеки файл трябва да **записва всички съобщения** вътрешно и да съхранява информация за нейния **размер**. Размерът се изчислява чрез **сумиране на ASCII кодовете** на всички букви от азбуката (a-Z) в текста. Например, appender със simple layout и съобщение **"3/31/2015 5:33:07 PM - ERROR - Error parsing request"** има размер **2606** (включвайки всички символи в PM, ERROR, Error, parsing, request). В случай на Xml layout файлът ще има размер **6632** поради **допълнителните символи** между таговете.

A file should write all messages internally and it should keep information about its size.

### Разширимост

Крайният потребител трябва да може да добавя собствени own **layout-и**/**appender-и**/**logger-и** и да ги използва. Например трябва да може да добави свой собствен **XmlLayout** и да направи така че appender-ите да го използват, **без директно да се редактира** кода на библиотеката.

|  |  |
| --- | --- |
| **Примерен код** | **Изход** |
| var xmlLayout = new XmlLayout();  var consoleAppender = new ConsoleAppender(xmlLayout);  var logger = new Logger(consoleAppender);  logger.Fatal("3/31/2015 5:23:54 PM", "mscorlib.dll does not respond");  logger.Critical("3/31/2015 5:23:54 PM", "No connection string found in App.config"); | <log>  <date>3/31/2015 5:23:54 PM</date>  <level>Fatal</level>  <message>mscorlib.dll does not respond</message>  </log>  <log>  <date>3/31/2015 5:23:54 PM</date>  <level>Critical</level>  <message>No connection string found in App.config</message>  </log> |

### Отчетен праг

Имплементирайте **отчетен праг** за всички appender-и. Appender-ът трябва да добавя само ссъобщения с отчетно ниво **по-високо или равно** на дадения праг (по **подразбиране** всички съобщения биват **добавени**). Отчетните нива са в следния ред: **Info** > **Warning** > **Error** > **Critical** > **Fatal**.

|  |  |
| --- | --- |
| **Sample Source Code** | **Output** |
| var simpleLayout = new SimpleLayout();  var consoleAppender = new ConsoleAppender(simpleLayout);  consoleAppender.ReportLevel = ReportLevel.Error;  var logger = new Logger(consoleAppender);  logger.Info("3/31/2015 5:33:07 PM", "Everything seems fine");  logger.Warning("3/31/2015 5:33:07 PM", "Warning: ping is too high - disconnect imminent");  logger.Error("3/31/2015 5:33:07 PM", "Error parsing request");  logger.Critical("3/31/2015 5:33:07 PM", "No connection string found in App.config");  logger.Fatal("3/31/2015 5:33:07 PM", "mscorlib.dll does not respond"); | 3/31/2015 5:33:07 PM - Error - Error parsing request  3/31/2015 5:33:07 PM - Critical - No connection string found in App.config  3/31/2015 5:33:07 PM - Fatal - mscorlib.dll does not respond |

Само съобщенията от Error нагоре са отпечатани.