DA1MCTA018-13SA - Programação Orientada a Objetos - Paulo Henrique Pisani - 2021.2

Descrição | Visualizar envios

[EP] Relatório com plugins

Data de entrega: segunda, 19 Jul 2021, 23:59
Arquivos requeridos: Relatorio.java, PluginTexto.java, PluginImpressao.java, RelatorioComPlugins.java, TextoTitulo.java, TextoData.java, ImpressaoSimples.java, ImpressaoLimiteLargura.java (Baixar)
Tipo de trabalho: Trabalho individual

Relatório com plugins

O diretor de uma empresa projetou uma classe para produzir relatórios. A classe trabalha com o conceito de plugins para processar o texto e para imprimir o texto de diferentes formas. Os plugins devem seguir uma interface predefinida. Dessa forma, se futuramente for necessário incluir um novo tipo de processamento de texto ou de impressão, bastaria implementar classes que sigam a interface para o plugin.

O diretor já implementou uma classe abstrata para o relatório e também duas interfaces (uma para plugin de texto e outra para plugin de impressão). Essa classe e as duas interfaces estão prontas e disponíveis na **aba Editar** (esses três arquivos não podem ser modificados).

- Classe abstrata Relatorio
- Interface PluginTexto
- Interface PluginImpressao

Entretanto, ainda falta implementar diversas partes para viabilizar o uso dessa classe para produzir relatórios com plugins. Neste exercício, deverão ser submetidos 5 arquivos (todas as classes descritas a seguir são públicas e devem estar no **pacote relatorio):**

- Classe RelatorioComPlugins
- Classes TextoTitulo e TextoData (implementações de plugin de processamento de texto)
- Classes ImpressaoSimples e ImpressaoLimiteLargura (implementações de plugin de impressão)

A seguir são apresentados os métodos necessários em cada classe. Esses métodos podem ser usados pelo sistema de correção e por isso devem seguir a especificação apresentada no enunciado. Você pode incluir atributos e métodos auxiliares adicionais que considerar necessários durante a implementação.

Classe RelatorioComPlugins

Essa classe deve ser subclasse da classe Relatorio e possuir os métodos a seguir:

- public RelatorioComPlugins(String autor): construtor que armazena o texto passado no parâmetro *autor*.
- public void aplicarPluginTexto(PluginTexto plugin): aplica o plugin de processamento sobre o texto armazenado na instância de RelatorioComPlugins e substitui o texto armazenado pelo obtido após a aplicação do plugin.
- public void setPluginImpressao(PluginImpressao p) : armazena uma referência a uma instância de um plugin de impressão. Essa instância será usada posteriormente pelo método imprimirRelatorio.
- public void imprimirRelatorio(): imprime o texto armazenado na instância de RelatorioComPlugins usando o plugin de impressão especificado por setPluginImpressao.

Plugins de processamento de texto

Os plugins de processamento de texto devem implementar a interface PluginTexto. Essa interface já está pronta e pode ser encontrada na aba Editar. Há apenas um método nessa interface:

- String aplicar(String texto): método que recebe um texto e retorna uma nova string com o resultado do processamento de texto aplicado. A regra de processamento de texto dependerá de cada classe que implementar este método.

Neste exercício, devem ser implementados dois plugins de processamento de texto:

Classe TextoTitulo: implementação de PluginTexto que adiciona um título ao texto do relatório.

- public TextoTitulo(String titulo): construtor que instancia um plugin e armazena um título na instância de TextoTitulo.
- public String aplicar(String texto): adiciona um título no início do texto no formato: "(TITULO: %s>) ", em que %s é o título do relatório (observe que não há acento). Por exemplo, para o texto "Este eh um teste." e título "Relatorio1", o retorno do método aplicar seria " (TITULO: Relatorio1) Este eh um teste."

Classe TextoData: implementação de PluginTexto que adiciona uma data ao texto do relatório.

- public TextoData(int dia, int mes, int ano): construtor que instancia um plugin e armazena a data (dia, mês e ano).
- public String aplicar(String texto): adiciona a data ao final do texto no formato " (DATA: %d/%d/%d)", em que cada um dos inteiros %d são respectivamente dia, mês e ano. Por exemplo, para o texto "Este eh um teste." e data 10/10/1000, o retorno do método aplicar seria "Este eh um teste. (DATA: 10/10/1000)".

Plugins de impressão

Os plugins de impressão devem implementar a interface PluginImpressao. Essa interface já está pronta e pode ser encontrada na aba Editar. Há apenas um método nessa interface:

- void imprimir(Relatorio r): método que recebe uma instância de Relatorio e imprime o texto do relatório (observe que RelatorioComPlugins é subclasse de Relatorio e, portanto, o argumento passado pode ser uma instância de RelatorioComPlugins). A regra de impressão dependerá de cada classe que implementar este método. **Importante**: para impressão use apenas o método **Impressao.imprimirLinha(String textoLinha).** A classe Impressao e este método imprimirLinha já existem no sistema de correção automática.

Neste exercício, devem ser implementados dois plugins de impressão:

Classe ImpressaoSimples: implementação de PluginImpressao que faz uma impressão simples.

- public ImpressaoSimples(): essa classe deve possuir apenas o construtor sem parâmetros (esse construtor é implicitamente criado se nenhum outro construtor for implementado na classe).
- void imprimir(Relatorio r): faz uma impressão simples, ou seja, apenas chama o método imprimirLinha com o texto do relatório.

Classe ImpressaoLimiteLargura: implementação de PluginImpressao que imprime o texto considerando que há uma largura limite para impressão.

- public ImpressaoLimiteLargura(int largura): construtor que armazena a largura máxima de linha passada no parâmetro largura.
- void imprimir(Relatorio r): imprime o texto considerando que há uma largura limite para impressão, ou seja, nenhuma linha impressa pode passar de um determinado número caracteres. Ao realizar esse tipo de impressão, algumas regras devem ser observadas:
  - as palavras não podem ser separadas. Portanto, se não tiver espaço disponível para incluir uma palavra em uma linha, ela deverá ser incluída na próxima. Por exemplo, para o texto "Frase teste" e largura de linha 8, a divisão seria em duas linhas "Frase" e "teste" (não há espaço na primeira linha para incluir "Frase teste". Se a largura de linha for 11, apenas uma linha é suficiente.
  - datas não podem ser separadas: a regra anterior se aplica a datas também (e.g. 10/10/1000).
  - toda linha deve iniciar com uma letra, ou seja, nenhuma linha deve iniciar com o caractere espaço ou com pontuação (ponto, vírgula, etc). Se algum sinal de pontuação ficar exatamente após uma quebra de linha, a palavra que precede o sinal deve ser passada para a próxima linha, de forma que nenhuma linha inicie com pontuação. Por exemplo, para o texto "Neste teste, a frase possui pontuacao." e largura de linha 11, a divisão seria em cinco linhas: "Neste", "teste, a", "frase", "possui", "pontuacao."
  - um caractere espaço que fique exatamente após uma quebra de linha deve ser suprimido, de forma que a próxima linha inicie com uma palavra (e não com um caractere espaço). Por exemplo, para o texto "Texto para teste ABCD" e largura 10, a divisão seria em duas linhas: "Texto para" e "teste ABCD".

Arquivos a serem submetidos

Neste exercício, deverão ser submetidos 5 arquivos:

- Classe RelatorioComPlugins
- Classes TextoTitulo e TextoData
- Classes ImpressaoSimples e ImpressaoLimiteLargura

Essa atividade disponibiliza também outros 3 arquivos com código pronto: Relatorio.java, PluginTexto.java e PluginImpressao.java. Não altere o conteúdo desses três arquivos.

Importante: Não é permitido o uso de expressão regular, assim como dos métodos indexOf, join, lastIndexOf, replace, replaceAll, replaceFirst, split, startsWith, trim. O programa principal já existe no sistema de correção automática. As classes submetidas não

podem realizar impressão de dados ou utilizar import **(apenas os plugins de impressão podem imprimir dados, mas devem usar Impressao.imprimirLinha(texto), método que já existe no sistema de correção)**.

Casos de teste

O programa de correção mantém um vetor com diversas instâncias de plugins e usa essas instâncias para realizar os testes (mais detalhes podem ser observados na sáida produzida pela correção automática). Formato dos casos de teste (que aparecem ao avaliar as classes no sistema de correção automática):

Entrada

- texto do relatório
- quantidade de plugins de processamento de texto a serem instanciados
- nomes das classes dos plugins de processamento de texto a serem instanciados (e seus parâmetros para inicialização)
- quantidade de plugins de impressão a serem instanciados
- nomes das classes dos plugins de impressão a serem instanciados (e seus parâmetros para inicialização)
- operações:
  - aplicarPluginTexto [índice plugin]: chama o método aplicarPluginTexto de RelatorioComPlugins e passa o plugin de texto especificado.
  - setPluginImpressao [índice plugin]: chama o método aplicarPluginTexto de RelatorioComPlugins e passa o plugin de impressão especificado.
  - imprimirRelatorio: chama o método imprimirRelatorio de Relatorio
  - aplicar [índice plugin] [texto]: chama o método aplicar de uma instância de plugin de processamento de texto (faz uma chamada direta ao plugin sem usar a classe RelatorioComPlugins).

Saída

- classes instanciadas
- métodos executados e saídas obtidas

Arquivos requeridos

Relatorio.java

```java
/*
 * IMPORTANTE: NAO ALTERE ESTE ARQUIVO
 */
package relatorio;

public abstract class Relatorio {
    private String autor;
    private String texto;

    public Relatorio(String autor) {
        this.autor = autor;
    }

    public String getAutor() {
        return this.autor;
    }

    public final void setTexto(String texto) {
        this.texto = texto;
    }

    public final String getTexto() {
        return this.texto;
    }

    public abstract void imprimirRelatorio();
}
```

PluginTexto.java

```java
/*
 * IMPORTANTE: NAO ALTERE ESTE ARQUIVO
 */
package relatorio;

public interface PluginTexto {
    String aplicar(String texto);
}
```

PluginImpressao.java

```java
/*
 * IMPORTANTE: NAO ALTERE ESTE ARQUIVO
 */
package relatorio;

public interface PluginImpressao {
    void imprimir(Relatorio r);
}
```

RelatorioComPlugins.java

TextoTitulo.java

TextoData.java

ImpressaoSimples.java

ImpressaoLimiteLargura.java

VPL

◄ [EP] Playlist

Seguir para...

[EP] REC - Arquivos multimídia ►