Introdução à Linguagem Java

Lista 05

19 de maio de 2021

Reutilização de Classes

Nome do projeto java: ListaX-NUSP. Sendo X o número da lista e NUSP o seu número usp.
Arquivo de envio: Um ÚNICO projeto java zipado, ou seja, enviar o arquivo ListaX-NUSP zipado.

Seguir as seguintes instruções, sob pena de desconto de nota por código confuso:

- Não enviar arquivos soltos.
- Não enviar mais de um projeto java.
- Não enviar um projeto com muitas subpastas fazendo com que dê trabalho para encontrar o exercício.
- Não enviar rascunhos dos exercícios. Envie apenas os arquivos que vocês efetivamente querem que sejam corrigidos para que não corram o risco de corrigirmos os arquivos errados.
- Colocar cada exercício dentro de um pacote diferente (o pacote fica dentro do projeto java). Exemplo: dentro do projeto Lista04-NUSP, inserir os pacotes ex1, ex2, ex3, ex4 e ex5. Dentro do pacote ex1 poderia haver os arquivos ContaCorrente.java e ContaCorrenteTeste.java.
- Separar a classe funcional da classe de testes deixando-as em arquivos diferentes.

Exercícios

1. Verifique a que a construção automática de classes derivadas funciona, através do seguinte exemplo:

```
class Base {
    Base() {
        System.out.println("Constrói Base");
    }
```

```
}
public class Derivada extends Base {
    Derivada () {
        System.out.println("Constrói Derivada");
    }
    public static void main(String []argc){
       Derivada obj = new Derivada();
    }
}
```

Adicione um parâmetro inteiro ao construtor da classe Base, o que deve gerar uma mensagem de erro. Insira um comentário explicando o erro. Corrija este erro sem alterar a classe Base.

2. Crie uma classe abstrata `NumeroAritmetico` com os seguintes elementos:
   - um atributo `protected long valor`.
   - um método concreto `public final boolean mesmoValor(NumeroAritmetico)`;
   - um método abstrato `someMeCom`, que recebe um `NumeroAritmetico` e devolve um `NumeroAritmetico`.
   - um método abstrato `subtraiaDeMim`, que recebe um `NumeroAritmeticoe` devolve um `NumeroAritmetico`.
   - um método abstrato `multipliqueMePor`, que recebe um `NumeroAritmeticoe` devolve um `NumeroAritmetico`.
   - um método abstrato `dividaMePor`, que recebe um `NumeroAritmeticoe` devolve um `NumeroAritmetico`.
3. Crie uma classe concreta `InteiroAritmetico` tal que:
   - estende a classe `NumeroAritmetico`;
   - crie um construtor público que recebe um parâmetro `long`.
   - implementa as operações aritméticas de maneira usual, devolvendo a si mesmo em cada caso;
   - Sobrecarrega o método `equals` da classe `Object`, que usa o método `mesmoValor` da classe mãe.
   - Sobrecarrega o método `toString` da classe `Object`.
   - Insira uma classe de testes que permita testar a funcionalidade dos métodos criados; se precisar, crie novos métodos públicos que permitam os testes.
4. Crie uma classe concreta `InteiroModular` tal que:
   - estende a classe `InteiroAritmetico`;
   - possui um segundo parâmetro `private long m`;
   - crie um construtor público que recebe dois parâmetros `long`, que chama o construtor `super`.

- implementa as operações aritméticas de modo que o resultadoseja sempre módulo `m` (o resto da divisão por `m`), devolvendo a si mesmo em cada casoe que utiliza a operação correspondente da classe mãe;
- Sobrecarrega o método `equals` da classe mãe, verificando a igualdade de todos os membros.
- Sobrecarrega o método `toString` da classe mãe.
- Insira uma classe de testes que permita testar a funcionalidade dos métodos criados; se precisar, crie novos métodos públicos que permitam os testes.

5. Crie uma classe concreta `InteiroModularComPotencia` tal que estende `InteiroModular` tal que:

   - Sua classe cont um método `elevadoA` que recebe um inteiro longo n e realiza a multiplicação do valor por si mesmo n vezes. Cuidado, o valor inicial deverá ser armazenado se o método `multipliqueMePor` for utilizado! O método devolve a si mesmo.
   - Teste o seu novo método 3 vezes. Em cada teste, o valor de m deve ser um inteiro **primo** maior que 10, e o `valor` deve ser outro inteiro maior que 1 e menor que m. Faça testes que verificam que

     $$a^m = a \pmod{m}, \text{ para } m \text{ primo}$$

     Este resultado é conhecido como *Pequeno Teorema de Fermat*, e é válido até mesmo se $a > m$. Verifique com um outro teste que o mesmo NÃO ocorre se m não é primo; idem para a não primo.

     Faça um teste que mostra que, entre a^1 e a^m, todos os diferentes valores de $a^j \pmod{m}$ se repetem o mesmo número de vezes, $1 \leq j \leq m-1$. Por exemplo, para $m = 13$ e $a = 3$, a sequência de portências de 3^1 a $3^{12} \pmod{13}$ é: 3,9,1,3,9,1,3,9,1,3,9,1. Note que cada um dos valores 3, 9 e 1 ocorrem 4 vezes (e sempre na mesma ordem!)

     Conclua que a sequência de potências na aritmética modular de primos é sempre cíclica e portanto uma boa forma de gerar distribuições uniformes de números inteiros pseudo-aleatórios.