Universidade Federal da Paraíba	Linguagem de Programação I
Centro de Informática	Semestre: 2017.2
Departamento de Informática	Professor: Derzu Omaia

2ª Lista de Exercícios

1. Considere um Sistema de Controle de Gastos de Mesada. Esse sistema apresenta uma classe chamada Despesa, com os atributos valor, do tipo double e tipoDeGasto, do tipo string e métodos para obter e alterar esses atributos (métodos get e set).
   Esse sistema apresenta também uma classe ControleDeGastos, que apresenta um atributo chamado despesas, que é um array de objetos do tipo Despesa e o método setDespesas. Essa classe apresenta também um método chamado calculaTotalDeDespesas, que não tem parâmetros e deve retornar o valor total das despesas do sistema. Nessa classe há ainda o método existeDespesaDoTipo(string tipoDeGasto) que verifica se dentre as despesas guardadas no ControleDeGastos há alguma delas que é do tipo passado retornando true neste caso e false, caso contrario.
   (a) Implemente a classe Despesa.
   (b) Implemente a classe ControleDeGastos.
   (c) Escreva um programa principal que crie um objeto do tipo ControleDeGastos e que tenha ao menos duas despesas. Faça com que o programa imprima o total das despesas utilizando o método correspondente da classe ControleDeGastos.

2. Considere um Sistema de Controle de Pagamentos de Funcionários de uma empresa. Esse sistema apresenta uma classe chamada Pagamento, com os atributos valorPagamento, do tipo double e nomeDoFuncionario, do tipo string e métodos para obter e alterar esses atributos (métodos get e set).
   Esse sistema apresenta também uma classe ControleDePagamentos, que apresenta um atributo chamado pagamentos, que é um array de objetos do tipo Pagamento e o método setPagamentos. Essa classe apresenta também um método chamado `calculaTotalDePagamentos`, que não tem parâmetros e deve retornar o valor total dos pagamentos do sistema. Nessa classe há ainda o método `existePagamentoParaFuncionario (string nomeFuncionario)` que verifica se dentre os pagamentos guardados no `ControleDePagamentos` há algum deles que se refere ao funcionário passado como parâmetro, retornando `true` neste caso e `false`, caso contrario.
   (a) Implemente a classe Pagamento.
   (b) Implemente a classe ControleDePagamentos.
   (c) Escreva um programa principal que crie um objeto do tipo ControleDePagamentos e que tenha ao menos dois pagamentos. Faça com que o programa imprima o total dos pagamentos utilizando o método correspondente da classe ControleDePagamentos.

3. Considere as duas classes abaixo:

```
class ClasseA {
    public:
    void metodoUm(int i) {
```

```
    }
    int metodoDois(int i) {
        return i*2;
    }
    int metodoTres(int i) {
        return i;
          }
    void metodoQuatro(int i) {
    }
}

class ClasseB : public ClassA {
    public:
    void metodoUm(int i) {
          }
    int metodoDois(int j) {
        return (j-1);
          }
    int metodoTres(int i, int j) {
        return i+j;
          }
    void metodoQuatro(double i) {
    }
}
```

a) Qual (is) método(s) sobrecarregam os métodos da superclasse?
b) Qual (is) método(s) sobrescrevem os métodos da superclasse?

4. Considere um programa que lida com Figuras Geométricas. Cada figura tem um nome e é capaz de calcular sua área. Para desenvolver esse programa, utilizou-se uma classe abstrata chamada `FiguraGeometrica`, com o atributo nome e com o método abstrato `calcularArea()`. Implemente essa classe e também algumas classes que herdem de `FiguraGeometrica` (ex: `Triangulo`, `Quadrado`, `Circulo`, etc). Em seguida, crie um programa (classe) principal para testar essas classes.

5. (a) Crie uma classe `Funcionario` com os atributos `matricula`, `nome` e `salário`, e os métodos `get()` e `set()` de cada atributo. Em seguida, crie uma classe `Consultor` que herda da classe `Funcionario` e sobrescreve o método `getSalario()`, adicionando um percentual de 10% no valor do salário. Implemente também um método `getSalario(float percentual)`, onde o parâmetro "`float percentual`" determina o percentual a ser acrescido no salário de `Consultor`.
   (b) Crie um programa (classe) principal `MinhaEmpresa` para testar as classes `Funcionarios` e `Consultor`. O programa deve criar um objeto da classe `Funcionarios` e `Consultor` e testar seus métodos.

6. a) Crie uma classe chamada `Data` para representar uma datas. Essa classe deve conter três atributos: o dia, o mês, e o ano. Considere também que a classe `Data` contém:
   - Um construtor que inicializa os três atributos e verifica a validade dos valores fornecidos;
   - Um método `set()` um `get()` para cada atributo;

- Um método `toString()` que retorna uma representação da data como uma string. Considere, nesse caso, que a data deve ser formatada mostrando o dia, o mês e o ano separados por barra (/).
- Um método `avancarDia()` que para avançar uma data para o dia seguinte.

b) Crie uma classe `DataTest`, com um método `main`, que cria alguns objetos da classe `Data` e utiliza as suas operações (métodos).

7. (a) Crie um programa para gerenciar um conjunto de médicos de um hospital. Deve ser criada uma superclasse `Medico` para a categoria geral de médicos e subclasses específicas para `Cirurgiao`, `Oftalmologista`, `Otorrino e Ginecologista`. Todos os médicos possuem um nome, uma altura e peso. Nas suas especialidades devem ter o nome do curso de especialização e um método que realize cirurgias ou atendimentos específicos para cada médico.
   (b) Crie um programa (classe) principal `Hospital` para testar as classes criadas acima. Para isso crie objetos de todas essas classes e teste os seus métodos.

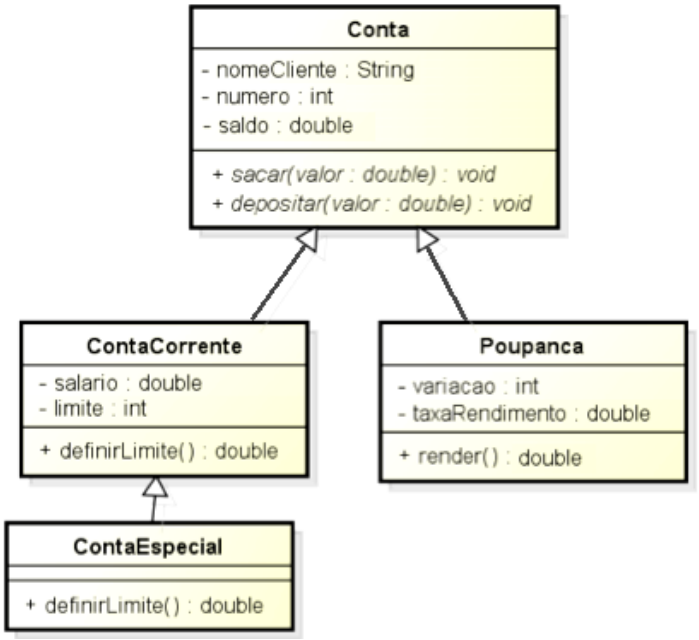
8. (a) Crie uma superclasse Trabalhador e subclasses TrabalhadorPorHora e TrabalhadorAssalariado. Cada trabalhador tem um nome e salário pago mensalmente.
- Escreva um método calcularPagamento (int Horas) que calcule o pagamento semanal de cada trabalhador.
- O trabalhador que ganha por hora é pago, obviamente, de acordo com o número real de horas trabalhadas, sendo horas, no máximo, igual a 40. Se ele trabalhou mais de 40 horas, cada hora excedente é paga como uma hora e meia.
- O trabalhador assalariado é pago pela carga horária de 40 horas e meia. O trabalhador assalariado é pago pela carga horária de 40 horas, independentemente de qual seja o número real de horas trabalhadas.
  (b) Crie um programa (classe) principal para testar as classes criadas.

9. (a) Crie uma classe abstrata pura (interface) `IConta` com os métodos "`void sacar(double valor)`" e "`void depositar(double valor)`".
   (b) Crie uma classe `Conta` que implementa `IConta` e que contenha os atributos `nomeCliente`, do tipo `string`, `salarioMensal`, `numeroConta`, `saldo` e `limite`, do tipo `double`, e os métodos para obter e alterar esses atributos (métodos `get` e `set`). Além disso, essa classe possui as seguintes características:
- Os valores dos atributos `nomeCliente`, `salarioMensal`, `numeroConta` e `saldo` são configurados no construtor da classe.
- O método `sacar` deve lançar uma exceção `SaldoNaoDisponivelException`, quando o valor a ser sacado é maior que o saldo disponível.
- O método "`void definirLimite()`", define o valor do atributo `limite` como 2 vezes o valor de `salarioMensal`.

(c) Crie uma classe `ContaEspecial` que herda da classe `Conta` e sobrescreve o método `definirLimite()` como 3 vezes o valor de `salarioMensal`.
(d) Escreva os códigos da classe de exceção `SaldoNaoDisponivelException` do exercício anterior.
(e) Crie a função main, o programa deve criar um objeto da classe `Conta`, chamar o método `sacar` (da classe `Conta`) e capturar a exceção `SaldoNaoDisponivelException`. Ao capturar a exceção, o programa deve imprimir uma mensagem informando o problema.

10. O que é polimorfismo? Dê um exemplo (com código) de um método polimórfico.

11. Em um grande banco existe um sistema de gerenciamento de contas. Considere que uma o seguinte diagrama de classes é uma versão simplificada dele.



Observações gerais: Os atributos herdados devem ser `protected` os demais `private`.

(a) (2,5) Crie os métodos `get` e `set` de todos os atributos de todas as classes. Crie também os construtores de todas as classes, estes devem receber como parâmetro todos os atributos da sua classe, e, caso herdem, os atributos do pai também. Nos construtores o construtor os atributos devem ser inicializados e, quando possível, o construtor do pai deve ser chamado.

(b) (2,5) Na classe `Conta` os métodos `sacar` e `depositar` devem alterar o valor do atributo saldo de forma apropriada. O método `sacar` deve possuir um `if` que exiba uma mensagem de saldo indisponível quando o valor a ser sacado é maior que o saldo disponível.

(c) (1,0) Na classe `Poupança`, o método `render` deve retornar o rendimento da poupança, baseado no `saldo` e na `taxaRendimento`. Se a variação da poupança for `51` o rendimento deve ser baseado apenas na taxa de rendimento, se for `1` a `taxaRendimento` deve ser acrescida de `0.5%`.

(d) (1,0) Na classes `ContaCorrente e ContaEspecial`, o método `definirLimite` deve retornar o duas vezes e quatro vezes, respectivamente, o salário do cliente.