**ATP**

O objetivo desta atividade prática (ATP) é construir um programa que faça uso adequado das principais técnicas de programação orientada a objetos para gerar a saída correta para toda e qualquer entrada de dados válida para o programa, isto é, que estejam no formato esperado pelo programa.  Este será implementado em quatro etapas, sendo melhorado gradativamente a cada uma delas com a aplicação das técnicas de orientação a objetos.  A cada etapa, você terá a oportunidade de colocar em prática as técnicas estudadas até então, podendo demonstrar o seu aprendizado para cada RA e ID da disciplina. Com isso, adquirirá competências que são fundamentais para o seu exercício profissional.

É fundamental que esteja atento aos requisitos apresentados em cada etapa e procure aplicar os conhecimentos vistos nas UAs da forma mais adequada, gerando um produto de qualidade, além de verificar o emprego das técnicas em uma aplicação muito próxima do que é feito no mundo real.

Observe que cada etapa é uma evolução da anterior, logo é fundamental cumprir bem cada uma para não comprometer o resultado do trabalho realizado na etapa seguinte.   Em caso de dúvida na realização de alguma tarefa, revise os conceitos vistos nas UAs e, se necessário, procure auxílio do professor- tutor para que o trabalho fique bem feito. Depois de pronta uma etapa, analise o trabalho realizado, verificando os benefícios das técnicas de programação orientada a objetos na resolução de problemas complexos.

O cumprimento dos prazos das etapas é fundamental para o bom andamento da atividade. Há um tempo estimado para a realização de cada uma, então programe-se  para não haver atrasos.

* O desenvolvimento integral da ATP prevê apenas uma entrega no fim da disciplina (semana 7 ou 8).
* A realização das etapas intermediárias de construção da atividade é fundamental, porém não prevê entregas isoladas.
* Embora as etapas não precisem ser entregues, você será acompanhado e supervisionado pelo professor-tutor no decorrer das semanas.
* Atente-se ao CHECKPOINT DA ATP na semana 4. Nessa semana, você responderá a questões referentes à(s) etapa(s) já realizada(s), as quais fornecerão dados que comprovam o cumprimento dos prazos e o alcance de resultados parciais.
* Explore efetivamente os canais de comunicação: fóruns, fale com o Prof-Tutor e sessões de webconferência. Sua participação ativa será considerada no momento de avaliação da atividade

**ETAPA 1**

Uma empresa pública possui um conjunto de funcionários de dois tipos:  concursados e temporários.  Para cada funcionário, independentemente do tipo, sabe-se o seu salário base.  Além disso, para cada funcionário concursado, sabe-se o número de anos   decorridos desde a sua contratação, enquanto, para cada funcionário temporário, o número de meses   total do período do seu contrato.

No fim de cada mês, a empresa emite um contra cheque para cada funcionário, contendo   o valor total recebido naquele mês. O valor mensal é dado pela soma do salário base   e da ratificação.  A gratificação de um funcionário concursado   é determinada pelo tempo de contratação:  ele recebe uma gratificação mensal de R$   200,00 para cada ano de contratação.   A de um funcionário temporário é determinada pelo período total do seu contrato:  ele recebe um a   gratificação   mensal de R$   15,00 para cada mês de contrato.

Implemente uma classe na linguagem Java, denominada FolhaPagamento ,  que contenha  apenas  o método  main ,   com os seguintes passos:

Lê os dados de n (entre 1 e 100) funcionários fornecidos pela entrada padrão (teclado).

Calcula o valor mensal recebido por cada funcionário.

Imprime na saída padrão (tela) a relação de n funcionários com os correspondentes valores mensais recebidos, apresentando primeiramente todos os concursados e, em seguida, os temporários.

Para cada funcionário, devem ser fornecidos os seguintes dados:

Código: um valor inteiro (int).

Tipo: o valor (int) 1 para funcionário concursado ou o valor 2 para funcionário temporário.

Salário-base: um valor real (double).

Tempo: um valor inteiro (int) correspondente ao número de anos de contratação para funcionário concursado ou ao número de meses do período de contrato para um funcionário temporário.

Por exemplo, considerando o seguinte conjunto de n = 3 funcionários:

Funcionário com código 147, concursado (1), com salário-base de R$ 4.000,00 e 5 anos de contratação.

Funcionário com código 304, temporário (2), com salário-base de R$ 1.500,00 e 24 meses de contrato.

Funcionário com código 214, concursado (1), com salário-base de R$ 7.200,00 e 10 anos de contratação.

Os correspondentes dados de entrada possuem o seguinte formato, sendo o primeiro dado   o valor de n:

3

147

1

4000.00

5

304

2

1500.00

24

214

1

7200.00

10

A saída impressa correspondente é:

Concursados:

147: 5000.00

214: 9200.00

Temporários:

304: 1860.00

Você construir a um programa que realiza as três etapas típicas, a saber:  entrada, processa mento e saída de dados, utilizando comandos sequenciais de uma linguagem de programação orientada a objetos, com expressões, repetição e decisão, e dados estruturados para armazenamento em memória.

Ao final, saberá codificar programas baseados em sequências de comandos imperativos e expressões sobre dados de tipos primitivos e estruturas de dados.

Para tanto, deverá considerar os comandos de definição de variáveis simples e estruturadas (vetores e matrizes), sintaxe para definição de expressões algébricas, comando de desvio, comando de repetição, comando para leitura de dados e comando para impressão de dados.

**ETAPA 2**

Reimplemente o programa feito na etapa 1, de forma a representar os dois tipos de funcionário por meio de classes específicas, denominadas FuncionarioConcursado e FuncionarioTemporario.

Cada uma dessas classes deverá definir atributos e métodos de acordo com as características do tipo de funcionário que representa. Assim, a classe  FuncionarioConcursado  deverá conter atributos que representem o código do funcionário, seu salário - base, o número de anos de contratação e o valor mensal recebido, enquanto a classe  FuncionarioTemporario   deverá conter  atributos que representem o código do funcionário, seu salário - base, o número de meses do contrato e o valor mensal recebido.

A classe FolhaPagamento continuará existindo, porém passará a fazer uso das novas classes para o armazenamento, o processamento e a impressão dos dados, isto é, deverá criar objetos dessas classes para representar os funcionários da empresa (um objeto para cada funcionário).

Cada uma das novas classes deverá ter um método construtor, um método que calcule o atributo correspondente ao valor mensal recebido pelo funcionário e um método que imprima os dados do funcionário (código e valor mensal recebido). Esses métodos deverão ser chamados a partir do método main da classe FolhaPagamento.

O conjunto de funcionários deverá ser representado por meio de dois atributos estáticos da classe FolhaPagamento, cada qual correspondente a um subconjunto dos funcionários: um atributo para o subconjunto dos funcionários concursados e outro atributo para o subconjunto dos funcionários temporários.  A implementação de cada subconjunto deverá ser feita com uso de alguma classe de biblioteca para armazenamento de coleção de objetos, como a classe ArrayList.

Você construirá um programa orientado a objetos, com o uso de classes e instanciação destas. Com isso, aprenderá a codificar classes com atributos e métodos, a criar objetos de uma classe, a codificar a interação entre objetos por meio de invocação de métodos, além de codificar operações sobre coleções de objetos.  Fique atento ao princípio de encapsulamento na implementação das classes.

Duas classes: classes específicas, denominadas FuncionarioConcursado e FuncionarioTemporario.

**ETAPA 3**

Reimplemente o programa feito na e tapa 2, com a introdução das  classes  Funcionario  e  Dependente .

A classe Funcionario deve ser criada como uma superclasse das classes FuncionarioConcursado e FuncionarioTemporario.   Dessa forma, deverá definir os atributos comuns às subclasses, enquanto cada subclasse deverá implementar a sua versão do método que calcula o valor mensal recebido, uma vez que as fórmulas de cálculo são diferentes para cada tipo de funcionário.

A classe Dependente deverá ser associada à classe Funcionario, de maneira que um funcionário possa ter de zero a cinco dependentes.  Tal associação deve ser implementada por um atributo da classe Funcionario, bem como por um método desta que permita ligar um dependente a um funcionário.   Ainda, a classe Dependente possui apenas um atributo do tipo inteiro para representar a idade (em anos) do dependente.

O  cálculo do valor mensal recebido por um funcionário deve levar em consideração  também  o seu conjunto de dependentes da seguinte maneira: um funcionário concursado tem um acréscimo de R$   100,00  para cada dependente  com idade igual ou inferior a  21  anos , enquanto um funcionário  temporário tem um acréscimo de R$   50,00 para cada dependente com idade igual ou inferior a  18  anos.   Para que isso seja realizado respeitando o princípio de encapsulamento, deverá ser definido um método na classe Funcionário que retorne o número de dependentes do funcionário que estejam no limite de idade (fornecido como parâmetro do método) para fins de acréscimo no valor mensal recebido.  Esse novo método deverá ser chamado pelos métodos das subclasses que calculam   o valor mensal recebido pelo funcionário.  Além disso, esse método da classe Funcionario deverá chamar um método da classe Dependente que recebe um valor inteiro como parâmetro e retorna verdadeiro quando a idade do funcionário é menor ou igual ao valor fornecido como parâmetro ou falso, caso contrário.

Para contemplar todas as modificações, o método  main  da classe  FolhaPagamento  deverá ser modificado de maneira que, para cada funcionário ,  seja definido o seu conjunto de dependentes , isto é, além de criar objetos das classes  FuncionarioConcursado  e  FuncionarioTemporario , o método  main  deverá criar objetos da classe  Dependente  e ligar cada dependente com o respectivo funcionário.  Isso implica que os dados de entrada sejam ampliados com os dados sobre os dependentes de cada funcionário.

Por exemplo, como feito nas e tapas 1 e 2, o funcionário com código 147, concursado (1), com salário-base de R$   4.000,00 e 5 anos de contratação tinha apenas os seguintes dados na entrada:

147

1

4000.00

5

Agora, se esse funcionário tiver dois ( 2 ) dependentes, um com  30  anos e outro com  18  anos, os dados de entrada  serão ampliados  da seguinte forma:

147

1

4000.00

5

2

30

18

Você construirá um programa orientado a objetos com o uso de herança entre classes e associação entre duas classes. Com isso, aprenderá a codificar classes genéricas e específicas e, ainda, aprenderá a fazer ligação entre dois objetos.   Observe as vantagens de criar classes genéricas, inclusive com respeito à definição e implementação de associação entre classes.

- Deverá ser definido um método na classe Funcionário que retorne o número de dependentes do funcionário que estejam no limite de idade. Esse novo método deverá ser chamado pelos métodos das subclasses que calculam   o valor mensal recebido pelo funcionário.

- Além disso, esse método da classe Funcionario deverá chamar um método da classe Dependente que recebe um valor inteiro como parâmetro e retorna verdadeiro quando a idade do funcionário é menor ou igual ao valor fornecido como parâmetro ou falso, caso contrário.

- Para contemplar todas as modificações, o método  main  da classe  FolhaPagamento  deverá ser modificado de maneira que, para cada funcionário ,  seja definido o seu conjunto de dependentes , isto é, além de criar objetos das classes  FuncionarioConcursado  e  FuncionarioTemporario , o método  main  deverá criar objetos da classe  Dependente  e ligar cada dependente com o respectivo funcionário.  Isso implica que os dados de entrada sejam ampliados com os dados sobre os dependentes de cada funcionário.

**ETAPA 4**

Reimplemente o programa feito na e tapa 3, tal que a classe FolhaPagamento passe a ter apenas um atributo para representar toda a coleção de funcionários. O tipo desse atributo deve ser alguma classe de biblioteca para a implementação de coleção de objetos, como ArrayList, sendo o tipo de objeto da coleção a superclasse Funcionario. Com isso, a coleção de objetos conterá tanto objetos da classe FuncionarioConcursado   quanto da classe FuncionarioTemporario, todos juntos.

Agora, no passo 2 do método  main , que calcula o valor mensal recebido por cada funcionário, deverá ser reimplementado como uma única iteração sobre a coleção de funcionários, chamando um método para o cálculo que esteja implementado na superclasse  Funcionario  como um método que simplesmente retorne  o valor zero e esteja sobrescrito em cada uma das superclasses apropriadamente, conforme a regra d e  cálculo do valor mensal recebido para cada tipo de funcionário.

Analogamente, no passo 3 do método  main ,   que imprime na saída padrão (tela) a relação de  n  funcionários com os correspondentes valores mensais recebidos, deverá haver apenas uma iteração sobre a coleção, chamando um método para impressão que esteja implementado na superclasse  Funcionario  como um método que imprima  todos os atributos genéricos do funcionário e esteja sobrescrito em cada uma das superclasses ,  a fim de imprimir também os atributos específicos apropriadamente. Neste caso, a implementação do método de impressão da subclasse deve chamar a implementação da superclasse.

Nesta nova versão do programa, a saída impressa será diferente da saída obtida nas etapas anteriores, que dividia a relação de funcionários em duas partes; os funcionários serão impressos de forma misturada.

Você construirá um programa orientado a objetos com o uso de herança entre classes e polimorfismo na implementação de coleções de objetos.   Assim, aprenderá a definir a versão genérica de um método e fazer sua sobrescrita em subclasses.  Observe que, apesar de chamar o método de forma genérica, o ambiente de execução do programa encontra e executa a versão específica adequada.