Revisão

Programação Orientada a Objetos

Nesta revisão, abordaremos os principais tópicos relativos ao conteúdo da disciplina de **Programação Orientada a Objetos**, para cada semana de aula que foi ministrada. Esta revisão deve ser utilizada como um guia para ajudá-lo a rever os principais assuntos abordados na disciplina. Espera-se que você tenha realizado as leituras sugeridas dos textos-base, feito as atividades propostas no decorrer de cada aula, bem como, sempre que possível, ter assistido aos vídeos de apoio. Foram disponibilizados códigos de exemplo que apareceram nas aulas, assim como códigos adicionais, para que pudesse utilizá-los para reforçar o aprendizado. Feito isso, tenho certeza de que terá sucesso na avaliação.

Na **primeira semana,** o objetivo foi apresentar alguns aspectos relacionados aos principais paradigmas de programação, com foco especial no paradigma orientado a objetos. Comentamos sobre o paradigma de programação estruturada, não estruturada, bem como o paradigma de POO, e tratamos de alguns frameworks de desenvolvimento para facilitar a criação de aplicações segundo o paradigma orientado a objetos. Destacamos também os conceitos de classes, objetos, atributos e comportamentos dos objetos e mensagens entre eles trocadas.

Na **segunda semana**, apresentamos os conceitos principais da disciplina: o paradigma de POO, que são fundamentais para aprender a programar utilizando os objetos, classes, herança, polimorfismo, encapsulamento etc. Uma classe é uma estrutura que tem a função de abstrair um conjunto de objetos com características similares. A classe define o comportamento dos objetos, por meio de métodos e os possíveis estados dos objetos, por meio dos seus atributos. O encapsulamento de dados de uma aplicação significa evitar que estes sofram acessos indevidos. Isso ocorre com a criação de uma estrutura que contém métodos que podem ser utilizados por qualquer outra classe. Apresentamos as diferenças entre classes e objetos, como lidar com atributos e métodos, bem como lidar com métodos especiais (acessadores, modificadores, construtores/destruidores). Utilizamos pacotes (conjuntos de classes relacionadas) para facilitar a localização e uso de tipos, evitar conflitos de nomes e realizar o controle de acesso, o que em geral deve ser considerado como uma boa prática para manter os códigos

organizados. Um bom material para leitura em que se discute algumas práticas para organizar classes em pacotes pode ser visto[1]. Finalmente discutimos métodos e variáveis estáticas.

Na **terceira semana,** apresentamos dois dos pilares da POO: a herança e o polimorfismo. O polimorfismo é uma característica relacionada à orientação a objetos que utiliza a hierarquia de objetos. Aprendemos também que a sobreposição de métodos ocorre quando há polimorfismo. Foram também discutidos termos como overload e override que, em geral, causam bastante confusão quando aprendemos POO. Além disso aprendemos que a herança permite o reuso de software ajudando a especializar soluções gerais já existentes. Finalmente abordamos o conceito de interfaces e sua aplicabilidade em orientação a objetos. As interfaces criam um contrato que deve ser obrigatoriamente obedecido pela classe que o implementa. No contexto de interface, quando programamos um software não importa como a implementação será feita, pois o fundamental é saber a definição do contrato, e garantir que o software desenvolvido por um grupo se "comunica" com outro grupo por meio deste contrato.

Na **quarta semana**, o assunto proposto foi sobre Generics, com foco na utilização de métodos (retornos e argumentos), bem como os principais usos de Generics em Java, destacando como declarar e instanciar um tipo genético. Trata-se de um conceito importante, uma vez que permite que tipos (classes e interfaces) sejam parâmetros na definição de classes, interfaces e métodos. Dentre as vantagens do Generics, temos a reutilização de códigos e evitar o uso de casting. Aprendemos também sobre coleções com a apresentação do framework Colletcions, pois este permite trabalhar com interfaces (definição abstrata de coleções), implementações (objetos concretos) e algoritmos (vários métodos para lidar com coleções). Com as Collections, utilizamos estruturas de dados existentes, sem nos preocuparmos com a maneira como são implementadas.

Na **quinta semana,** aprendemos sobre conjuntos de classes relacionadas que são organizadas em pacotes com o objetivo de facilitar a localização e uso de tipos, evitar conflitos de nomes e fazer controle de acesso. Os conjuntos são coleções que não permitem elementos duplicados, sendo representado pela interface Set, tendo como principais implementações as classes HashSet,

1 http://www.usp.br/thienne/coo/material/aula1-pacotesjava

LinkedHashSet e TreeSet. Destacamos também a interface Map e suas classes implementadoras e conhecemos como funcionam os principais métodos e o processo de desenvolvimento para ordenar um mapa. Vimos que o uso de Mapas é muito interessante quando queremos buscar um objeto, considerando que temos alguma informação sobre ele. Um mapa é composto por um conjunto de associações entre um objeto chave e um objeto valor, sendo equivalente ao conceito de dicionário, presente em linguagens de programação.

Na **sexta semana,** abordamos os conceitos que envolvem as exceções, que são geradas em qualquer código a ser executado em um computador. Quando criamos programas em Java, erros imprevistos podem ocorrer durante as execuções. Exceções acontecem quando por exemplo tentamos abrir um arquivo inexistente, tentamos conectar a um servidor que está fora do ar etc. ou erros de lógica (divisão de um número por zero e utilização de classes ou métodos inexistentes). Sempre que um erro ocorre, uma exceção (objeto) é criada e lançada. A exceção por sua vez encapsula as informações do erro e essa exceção deve ser capturada em algum momento. Discutimos também os tipos de exceções checadas e não checadas e como trabalhar com elas no contexto da disciplina.

Para finalizar, chegamos à **última semana** do curso, em que discutimos o mecanismo de serialização de objetos para representar o estado de um objeto como uma sequência de bytes. Apresentamos o conceito de Streams e a API Stream da linguagem Java e sua utilização na Programação Orientada a Objetos (POO). Streams representam o fluxo contínuo de dados de entrada e saída e pode estar associado a fontes como, arquivos, conexão de rede, outros programas, entrada (teclado), memória etc. Baseiam-se no fluxo unidirecional de dados, podem ser inclusive de tipos diferentes: primitivos (char, byte) e objetos. Outro assunto da semana foram as threads e seus recursos, e alguns conceitos importantes relacionados ao sistema operacional, como time slicing e processos. Threads são chamados de processos leves, e o principal motivo para utilizá-las é fazer como que o sistema operacional consiga dividir as tarefas entre todos os processadores disponíveis, o que tende a aumentar a eficiência geral do sistema, bem como simplificar a modelagem de algumas aplicações. Discutimos também como desenvolver aplicações em Java que podem comunicar-se via rede local ou rede remota (Internet),

usando sockets. Os sockets são compostos por um conjunto de primitivas do sistema operacional e foram originalmente desenvolvidos para o BSD Unix.

De um modo geral, uma sugestão para todos vocês é revisar as atividades avaliativas, bem como os textos-base, códigos disponibilizados, vídeos e ferramentas que foram solicitadas que vocês interagissem durante as semanas. Tais materiais contém informações importantes cujo entendimento é primordial e que podem aparecer na prova.

Como o docente responsável pela disciplina, espero que os assuntos tratados possam complementar a sua formação profissional e que você tenha sucesso no decorrer da sua carreira, que está apenas começando. Tenham em mente que todo o esforço será recompensado.

Desejo sucesso a todos e uma boa prova!

Prof. Dr. Julio Cezar Estrella