Objetivos	a. Demonstrar o funcionamento de diversas ED sob os aspectos: ocupação de memória, armazenamento e recuperação de conteúdo, operações principais b. Estabelecer a ligação entre o mundo "estruturado-procedimental" (linguagem C) e o mundo "orientado a objetos" (Java)
Referências gerais	DEITEL, H.M.; DEITEL, P.J. *Java: How to Program*. 6th ed. Upper Saddle River (NJ): Pearson, 2005. W3 SCHOOLS. *Java Tutorial*. Disponível em <https://www.w3schools.com/java/default.asp>. Acesso em 08 mar 2023. DEV.JAVA. *Learn Java*. Disponível em <https://dev.java/learn/>. Acesso em 08 mar 2023.

Objetivo específico desta aula prática: apresentar conceitos de **composição** e **herança** em Java, cuja finalidade é permitir o reaproveitamento de código.

Suporte teórico 1. **Composição** em Java. Deve ser utilizada sempre que se entende que um objeto de uma classe tem um objeto de outra classe.

O que	Sintaxe Java	Obs.
A composição caracteriza-se pela utilização, na definição de uma classe, de um atributo que é na verdade um objeto de uma outra classe ("componente"). Assim, para esse atributo estão automaticamente disponíveis todos os atributos e métodos definidos para a classe "componente".	// definição de classe componente public class ClasseComponente { // definição de atributos e métodos … } // definição de classe composta public class ClasseComposta { // atributo objeto da componente ClasseComponente atrib; // outros atributos e métodos próprios … }	No exemplo, o atributo atrib será instanciado, dando origem a um objeto da classe componente. Se ao atributo atrib especificar-se acesso do tipo private, mesmo os itens públicos da classe componente não estarão visíveis fora da classe composta.
Exemplo prático 1 Todo empregado tem um nome.	// definição de classe composta public class Empregado { // atributo objeto da componente String nome; // outros atributos e métodos próprios … }	Objetos da classe String (a qual é "nativa" no Java, mas não primitiva) são frequentemente utilizados como componentes.
Exemplo prático 2 Todo cliente tem um endereço.	// definição de classe componente public class Endereco { // definição de atributos e métodos … } // definição de classe composta public class Cliente { // atributo objeto da componente Endereco enderecoEntrega; // outros atributos e métodos próprios … }	A classe componente deve estar "visível" para a classe composta, seja no mesmo arquivo, no package ou ainda por meio de import.

Continua ...

Suporte teórico 2. **Herança** em Java. Deve ser utilizada sempre que se entende que um objeto de uma classe é um objeto de outra classe (mesmo que parcialmente).

O que	Sintaxe C#	Obs.
A herança permite que atributos e métodos escritos para uma classe ("super', "base" ou "mãe") estejam automaticamente disponíveis para outras classes ("sub", "derivadas" ou "filhas"). Estas últimas podem estender a definição da classe base, renomeando, aumentando ou especializando seus atributos e métodos.	*// definição de classe base* public class ClasseBase { *// definição de atributos e métodos* … } *// definição de classe derivada* public class ClasseDerivada extends Classe_Base{ *// atributos próprios* … *// método construtor* public ClasseDerivada (…) { *// aciona construtor da classe base* super(…); *// trata atributos próprios* super(…); } *// outros métodos próprios* … }	Na definição da classe derivada, a palavra reservada extends indica a herança. Os atributos e métodos da classe base são herdados, com exceção do construtor e de itens especificados como private na classe base. Os especificadores de acesso dos itens da classe base se mantêm os mesmos na herança, isto é, serão os mesmos nas classes derivadas.
Exemplo prático 1 Em Java, qualquer classe é um objeto da classe Object.	*// definição de classe composta* public class Empregado extends Object{ *// atributos e métodos próprios* … *// método que sobrescreve 'toString'* public String toString() { … } }	O método toString é definido na classe Object e pode ser sobrescrito ("redefinido") em qualquer outra classe para formatar a apresentação dos atributos desta classe.
Exemplo prático 2 Na Fatec, qualquer professor é um servidor público.	*// definição de classe base* public class ServidorPublico { *// definição de atributos e métodos* *// comuns a qualquer servidor público* … } *// definição de classe composta* public class Professor extends ServidorPublico { *// atributos específicos de cada professor* … *// método construtor* public Professor (…) { *// aciona construtor da classe base* super(…); *// trata atributos específicos* super(…); } *// métodos específicos de cada professor* … *// eventuais métodos que sobrescrevem* *// outros da classe base* … }	Uma classe derivada, por sua vez, pode ser a classe base de outras, estendendo a linhagem por meio de herança. Por exemplo, a classe Professor pode ser base de ProfessorHorista e ProfessorMensalista, por dentre outras possibilidades. Se uma classe deve ser a última "herdeira", interrompendo a possibilidade de novas heranças, ela precisa ser especificada como final.

Atividade 1. Escreva, em linguagem Java, um projeto para prencher e apresentar dados de dois tipos de empregado: aquele que recebe como pagamento (unicamente) um valor calculado diretamente a partir de suas vendas (empregado *comissionado*) e aquele que, além de comissão, recebe também um valor fixo (empregado *comissionado mais fixo*). Estabeleça entre esses dois tipos de empregado um relacionamento de **herança**. Sugestão: utilize como ponto de partida os códigos das classes EmpregadoComissionado.java e EmpregadoComissionadoMaisFixo.java disponíveis no ambiente *online* da disciplina.

Atividade 2. Inclua, na(s) classe(s) base(s) do projeto acima, o atributo dataNascimento (que armazena o dia, mês e ano de nascimento do empregado). Para isto, recorra ao mecanismo de **composição**. Certifique-se de que todas as exibições de dados de empregado apresentem agora o conteúdo do respectivo atributo dataNascimento. Sugestão: utilize como ponto de partida o código da classe DiaMesAno.java, disponível no ambiente *online* da disciplina.

Desafio. Reescreva o projeto da Atividade 1, porém supondo que o relacionamento entre as classes que representam o empregado *comissionado* e o empregado *comissionado mais fixo* é de **composição**, isto é, EmpregadoComissionado é componente de EmpregadoComissionadoMaisFixo.