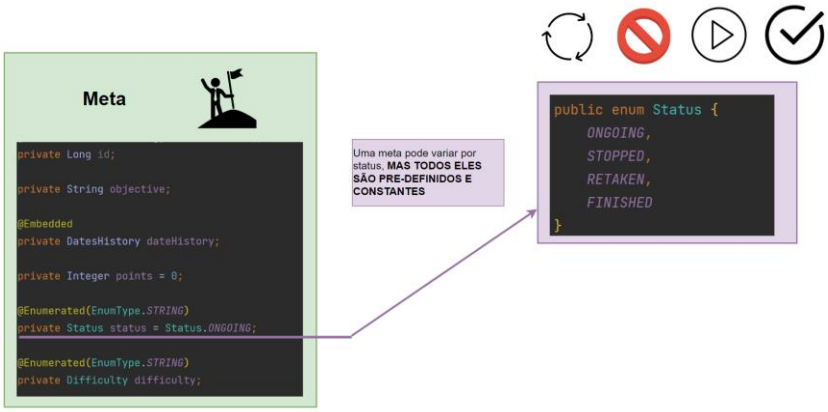
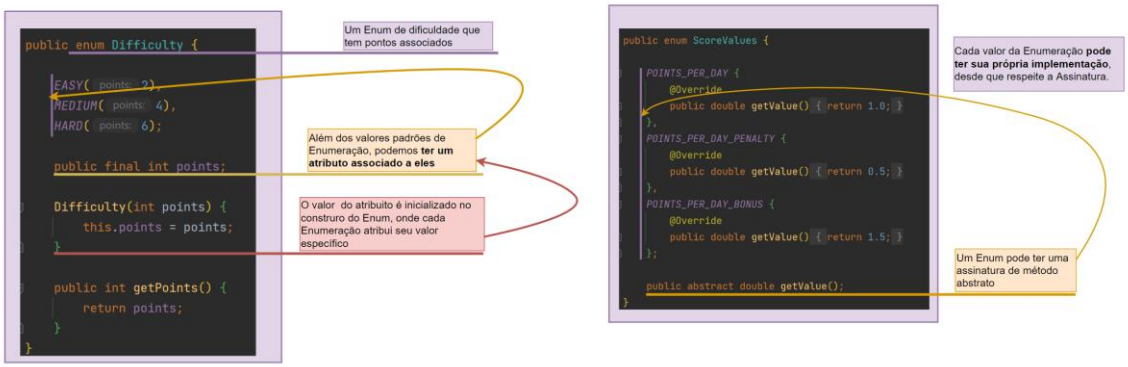
Enums – Java

Enums são um tipo de classe que representa **valores CONSTANTES** (que não mudam). São interessantes de se utilizar quando você tem uma quantidade de possibilidades pré-definidas para se trabalhar.

Um uso comum do Enum é **representar o estado** de um determinado Objeto:



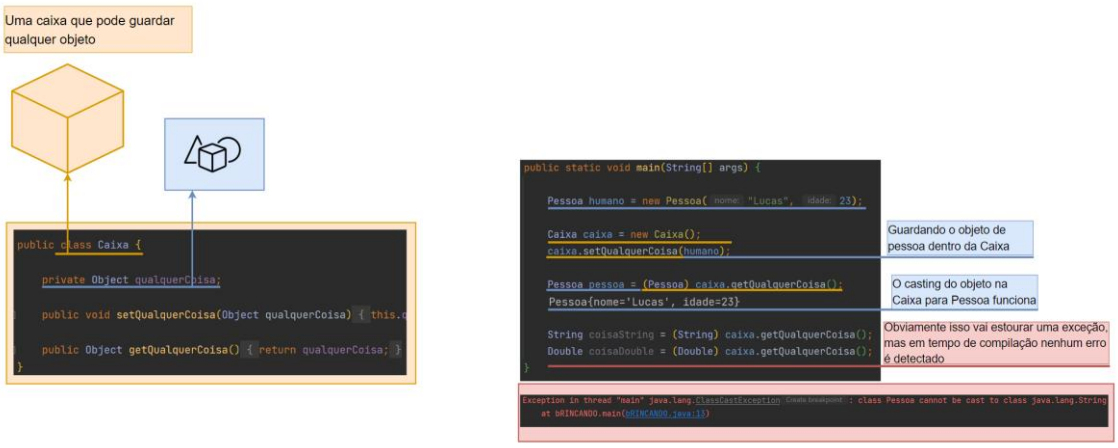
Embora o **Enum** possa representar estados, ele também é capaz de retornar valores específicos e ter comportamentos:



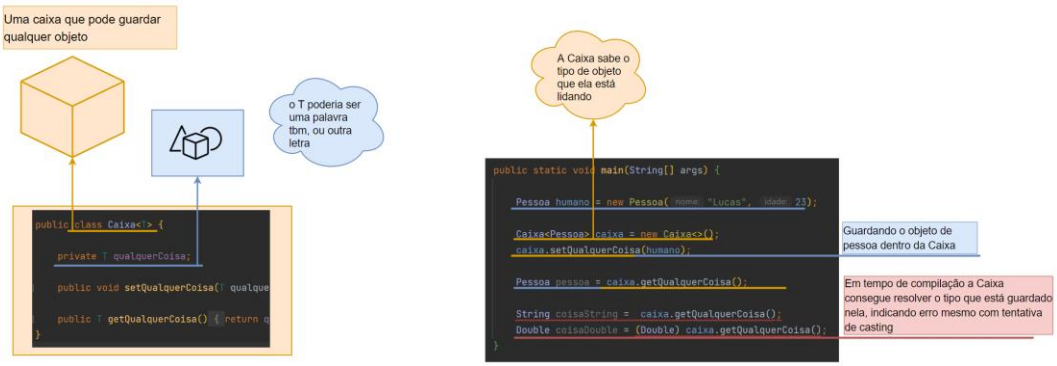
Generics – Java

Generics trazem benefício de **type safety**. Por exemplo, se eu fizesse uma abstração **usando o tipo Object**, eu precisaria implicitamente **fazer o casting pro tipo específico quando eu precisasse**, isso

poderia lançar uma **RuntimeException**:



Quando usamos Generics garantimos uma segurança de erro a nível de compilação, isso porque "resolvemos" o tipo específico do genérico quando instanciamos a classe:



Generics Delimitados – Java

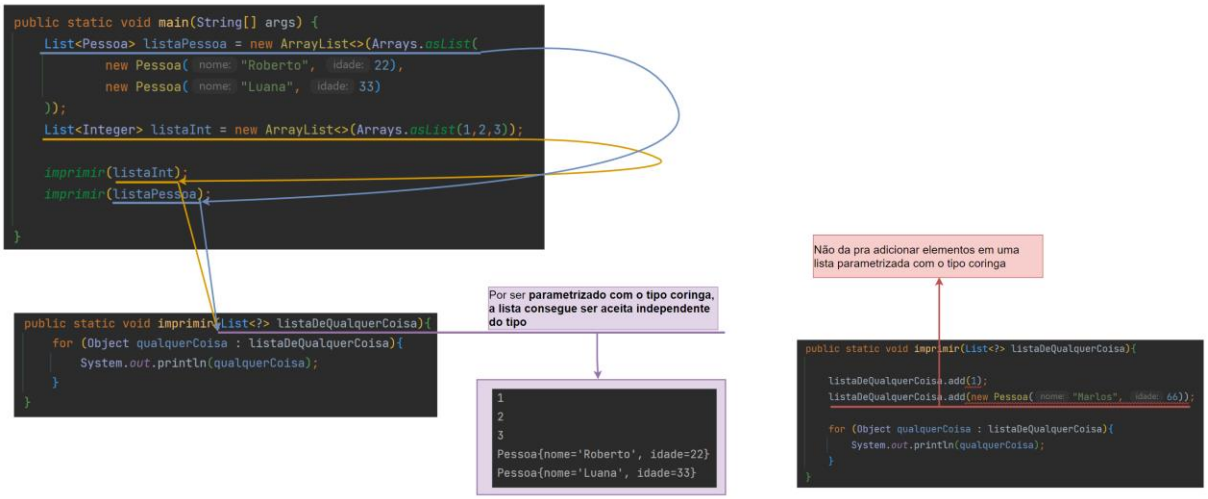
Os **Generics** tem bem mais poderes do que só facilitar a vida do desenvolvedor em tempo de compilação (o que já é muito). Você pode aliar o uso de genéricos com herança para garantir coesão

de uma classe:

Isso também é util quando você itera sobre uma lista genérica e precisar usar comparação por exemplo, você pode delimitar que o Objeto genérico "extends" de Comparable.

Coringa (wildcard Types) – Java

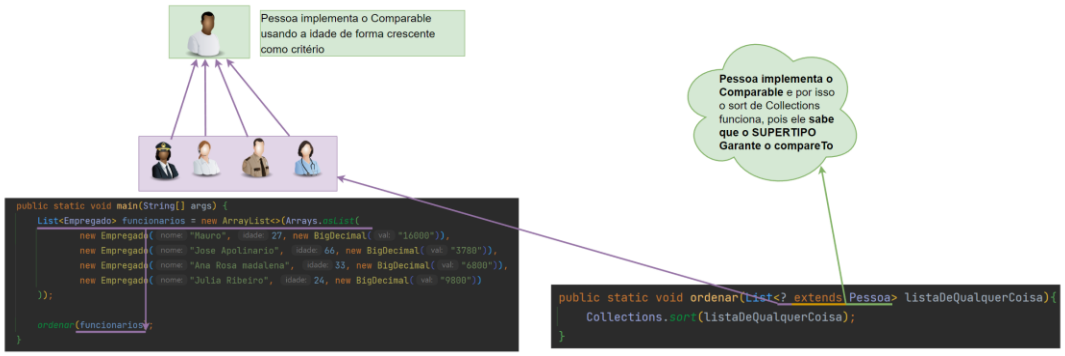
O tipo coringa é representado por **“<?>”.** Os tipos coringa são **usados como referência apenas**, por exemplo, e**u posso imprimir uma lista coringa**, mas **não posso adicionar nela**:



Coringas/Generics delimitados com super e extend – Java

Generico/Coringa que Extende um super tipo:

Quando especificado que o **coringa EXTEND** de **algo**, você está dizendo que pode acessar esse **coringa** e usar os **“métodos”** dele que são compatíveis com o **“algo”**:

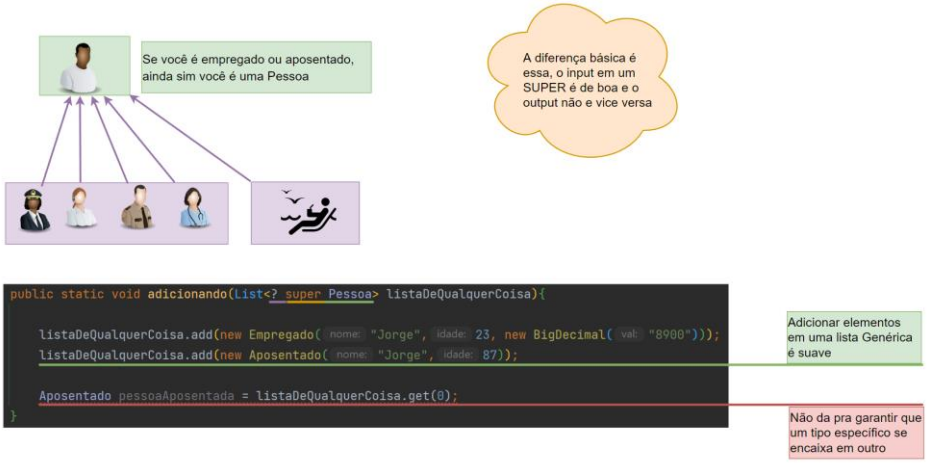


Não dá para adicionar em uma lista coringa que EXTEND de um supertipo, **isso porque um subtipo pode ser diferente de outro**, então você não tem garantia que os subtipos sejam compatíveis.

Generico/Coringa que tem um super de determinado Tipo:

Aqui eu posso adicionar subtipos diferentes, isso porque tanto faz desde que o Super deles sejam o mesmo. Mas eu não posso recuperar esses elementos, porque eu não posso garantir que eles sejam

compatíveis a nível de instância:



Num macro a diferença básica é essa, um <?> que extende pode ser acessado, um <?> que tem um super pode ser adicionado. Ai dá pra combinar os dois em algum trabalho ultra genérico.

Coringas/Generics diferença for real – Java

O **coringa <?> você não tem noção do que pode estar parametrizado al**i, então você tem menos opções de manipulação, como adicionar em uma lisa.

Usando **Generics você está especificando um tipo quando for usar a classe**, então você tem mais opções de manipulação, pois você pode garantir que o tipo genérico vai ser mais “controlável”.