22. Třídy a objekty – rozhraní (Java)

Rozhraní (Interface)

Rozhraní v Javě umožňuje definovat společné vlastnosti pro skupinu tříd a specifikovat metody, které budou moci pracovat s libovolnou instancí této skupiny. Lze ho chápat jako smlouvu mezi rozhraním a třídami, které ho implementují. Umožňuje definovat společné chování, ale neobsahuje konkrétní implementace. Změny v implementaci nemají vliv na ostatní kód.

Implementujeme pomocí klíčového slova – implements

Metody

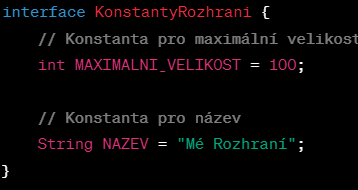
Deklaruje abstraktní metody (metody bez implementace), které třídy implementující rozhraní musí poskytnout vlastní implementaci.

Dále také definuje metody s implementací (defaultní metody), které nemusí být povinné pro implementující třídy.

Obsah obrázku text, snímek obrazovky, Písmo

Popis byl vytvořen automaticky

Konstanty

Jsou proměnné, které nemohou být změněny po svém přiřazení. Konstanty v rozhraní jsou implicitně veřejné, statické a final, a proto se často označují jako "konstanty rozhraní". 

Je k nim přistupováno přímo pomocí jména rozhraní bez vytváření instance, např. **KonstantyRozhrani.MAXIMALNI\_VELIKOST**. Tyto konstanty jsou využívány pro definici hodnot, které by měly být společné pro všechny třídy implementující dané rozhraní.

Implementace rozhraní

Obsah obrázku text, snímek obrazovky, Písmo

Popis byl vytvořen automaticky

Třída může implementovat více než jedno rozhraní, což umožňuje dosáhnout víceúrovňové dědičnosti.

Obsah obrázku Písmo, snímek obrazovky, text, Grafika

Popis byl vytvořen automaticky

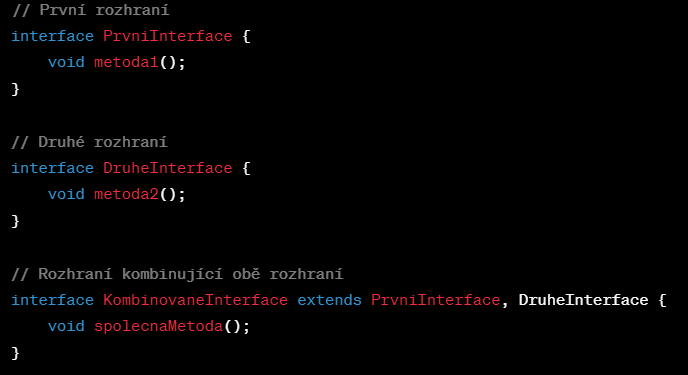
Dědění

V Javě může rozhraní dědit od jiného rozhraní, což umožňuje sdílet metody a konstanty mezi více rozhraními. Stejně jako u tříd se k dědění používá klíčové slovo extends.

1. Jednoduché děděníObsah obrázku text, snímek obrazovky, Písmo

   Popis byl vytvořen automaticky

**DedicInterface** bude obsahovat všechny tři metody.

1. Více dědičnost

Diamantový problém nemůže nastat u abstraktních metod a konstant. Ale defaultní metody (metody s implementací) mohou vytvářet konflikty. Jak u dědění, tak u implementace.

Konflikty lze řešit:

//obě rozhraní mají defaultní metodu nejakaMetoda

1. Přepsáním metody (@override)Obsah obrázku text, snímek obrazovky, Písmo

   Popis byl vytvořen automaticky
2. Explicitní volání jedné z metod

Obsah obrázku text, snímek obrazovky, Písmo

Popis byl vytvořen automaticky

Abstraktní třídy

//podle Kubinové to patří tady a ne do dědičnosti.

Je třída, která nemůže být přímo instanciována (nemůžete vytvořit objekt této třídy) a slouží k definování obecných vlastností a metod pro její potomky.

Abstraktní třída je deklarujeme klíčovým slovem **abstract**

Může obsahovat jak konkrétní metody s implementací, tak i abstraktní metody, které jsou deklarovány, ale nemají definovanou implementaci. Potomci abstraktní třídy musí implementovat všechny její abstraktní metody.

Obsah obrázku text, snímek obrazovky, Písmo

Popis byl vytvořen automaticky