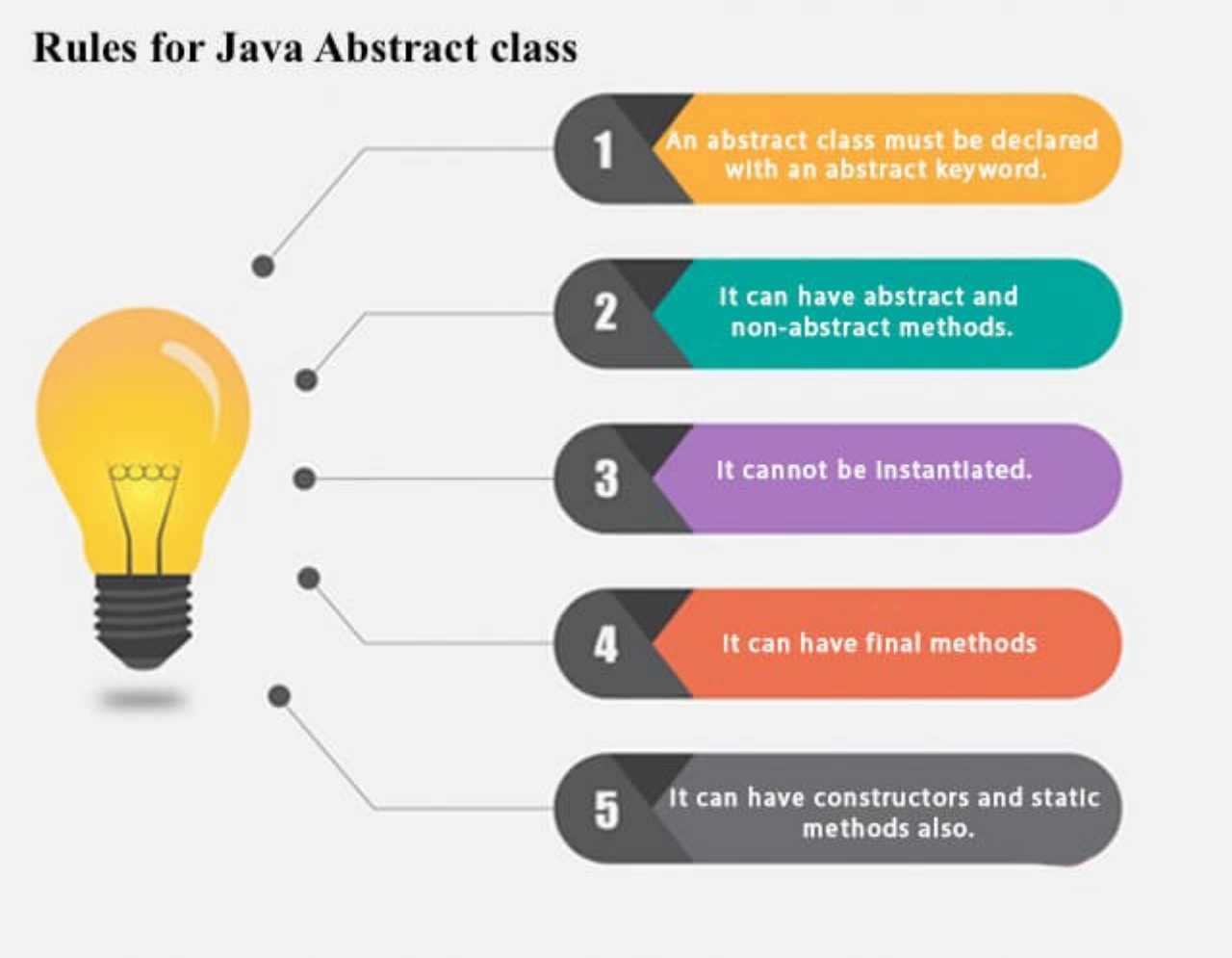
Abstraktná trieda

Trieda, ktorá obsahuje jednu alebo viac abstraktných metód, sa nazýva abstraktná trieda a trieda je deklarovaná ako abstraktná pomocou kľúčového slova "abstrakt", pred ktorým je kľúčové slovo "trieda" na začiatku deklarácie triedy. Abstraktná trieda obsahuje abstraktnú metódu, ktorú tvorí neúplný typ. Preto nemôžete vytvárať objekty (inštancie) abstraktnej triedy. Kedykoľvek trieda zdedí abstraktnú triedu, musí implementovať všetky abstraktné metódy abstraktnej triedy, ak to nie je, potom musí byť tiež deklarovaná ako abstraktná. Abstraktný atribút je dedičný, kým sa nedosiahne úplná implementácia abstraktných metód.

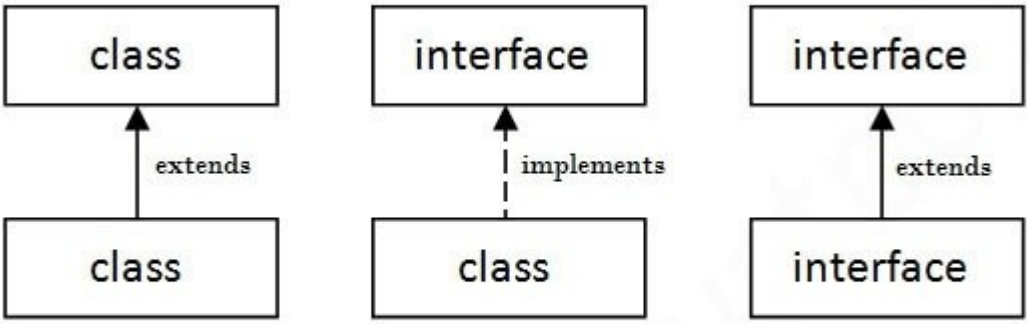
Abstraktná trieda môže obsahovať aj konkrétne metódy, ktoré môžu byť použité odvodenou triedou.



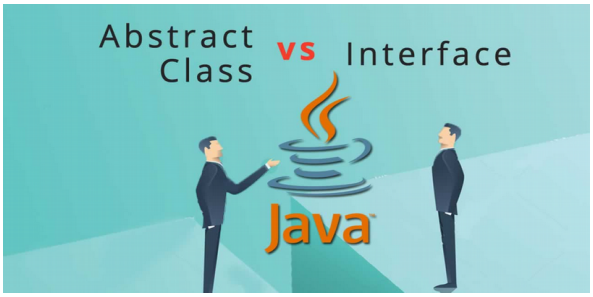
Rozhranie

Rozhranie je čisto abstraktná trieda, všetky metódy v rozhraní sú úplne abstraktné. Rozhranie špecifikuje len to, čo trieda musí robiť, ale nedefinuje, ako to robí.

- Premenné rozhrania sú štandardne vždy verejné, statické a konečné (final).
- Premenné sa musia iniciovať v čase deklarácie.
- Premenné nemožno nikdy nastaviť ako súkromné, chránené, prechodné (non-final) a nestatické (non-static).
- Metódy rozhrania sú vždy verejné a abstraktné. Nemôžu byť súkromné, chránené, konečné, statické.
- Nemôžeme deklarovať žiadny konštruktor vo vnútri rozhrania, pretože hlavným účelom konštruktora je inicializácia triednych premenných, ale v rozhraní sú premenné inicializované v čase deklarovania.
- Rozhranie môže dediť z iných rozhraní, ale trieda implementujúca takéto rozhranie musí implementovať metódy všetkých zdedených rozhraní.
- Trieda môže zdediť viac ako jedno rozhranie naraz a musí implementovať všetky metódy všetkých zdedených rozhraní.

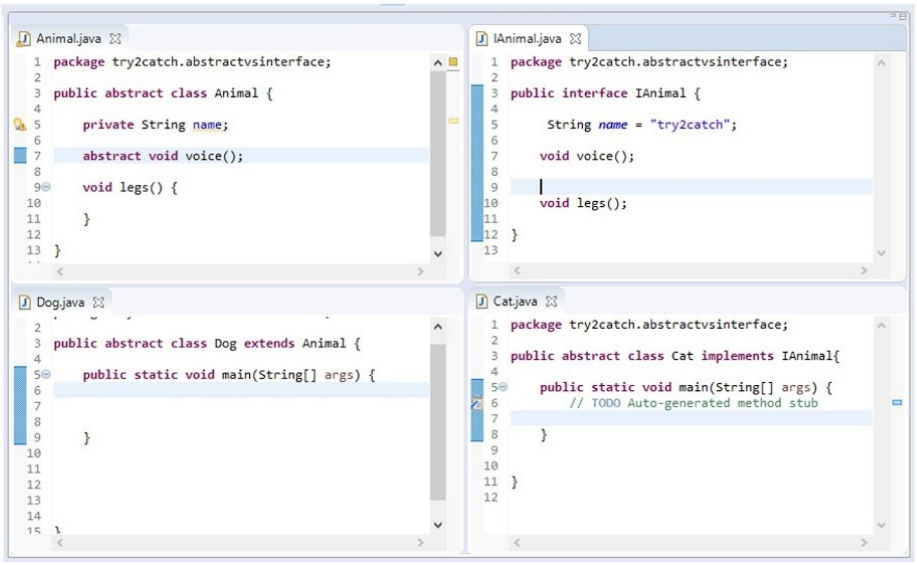


Multiple Inheritance in Java



Základ pre porovnanie	Rozhranie	Abstraktná trieda
Metódy	Rozhranie obsahuje iba abstraktné metódy.	Abstraktná trieda obsahuje abstraktné metódy (bez tela), ale okrem toho môže mať implementované aj statické alebo nestatické metódy (s telom).
Premenné	Premenné rozhrania nie je možné označiť za súkromné, chránené. Rozhranie môže mať len konštanty (final), public a static. Implicitne sú všetky premenné rozhrania nastavené na final. Musia sa inicializovať pri deklarácii.	Neexistujú žiadne obmedzenia pre premenné, môžu byť final, static a aj klasické premenné – private, public, ... Nie je potrebné ich inicializovať pri deklarácii.
Modifikátor prístupu metód	Metódy rozhrania sú vždy "verejné" a "abstraktné", aj keď to neuvedieme. Preto môžeme povedať, že interface je 100% čistá abstraktná trieda.	Nie je povinné, aby metóda v abstraktnej triede bola verejná a abstraktná. Môže mať aj konkrétne metódy.
Konštruktory	Konštruktor nemôžeme deklarovať vo vnútri rozhrania.	Abstraktná trieda môže mať konštruktor.
Dedičnosť	Rozhranie môže dediť len z rozhrania.	Môže dediť aj z inej triedy aj implementovať rozhranie.
Kedy použiť	Keď potrebujeme viacnásobnú dedičnosť alebo keď chceme urobiť úplnú abstrakciu – prepíšeme len aké metódy musia obsahovať potomkovia, implementácia je už na potomkoch.	Keď chceme, aby potomkovia používali rovnakú časť kódu a ostatné metódy mali predpísané.

Java Abstract Vs Interface



Animal.java

```
package try2catch.abstractvsinterface;

public abstract class Animal {

    private String name;

    abstract void voice();

    void legs() {

    }
}
```

IAnimal.java

```
package try2catch.abstractvsinterface;

public interface IAnimal {

     String name = "try2catch";

    void voice();

    void legs();
}
```

Dog.java

```
public abstract class Dog extends Animal {

    public static void main(String[] args) {



    }
```

Cat.java

```
package try2catch.abstractvsinterface;

public abstract class Cat implements IAnimal{

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

	Interface	Abstract Class
Constructors	✗	✓
Static Fields	✓	✓
Non-static Fields	✗	✓
Final Fields	✓	✓
Non-final Fields	✗	✓
Private Fields & Methods	✗	✓
Protected Fields & Methods	✗	✓
Public Fields & Methods	✓	✓
Abstract methods	✓	✓
Static Methods	✓	✓
Final Methods	✗	✓
Non-final Methods	✓	✓
Default Methods	✓	✗