* **Анонимни класове** - Вид **вътрешни класове**, но **без имена**. Анонимните класове **не могат да дефинират в своето тяло конструктори** и за инициализация използват конструкторите на предшественика си.
* **Идентификация по време на изпълнения** (Run-time Type Identification - **RTTI**) - Идентификацията на типове по време на изпълнение на програмата е мощен механизъм, който позволява да разберем типа на дадена инстанция, имайки само референцията към нейния базов клас.
* Оператор **instanceof**
* **RTTI техниката се базира на специализирания обект от тип Class, в който Java съхранява метаинформация за съответния тип клас**. Когато извикаме оператора "instanceof", той използва метаданните, представяни от посочения от нас обект и клас, за да се провери израза.

**instanceof**vs. **Class.isAssignableFrom()** - Използването на instanceof е по-добър и по-елегантен подход

* **Преобразуване на един референтен тип към друг референтен тип (кастване).**

**Композиция**- Създаването на обект в контекста на даден друг клас

* **Enumeration**(Изброен тип) - Конструкция, наподобяваща клас, в която можем да декларираме само логически свързани константи.

**Вътрешни класове**- Езикът Java позволява създаването на дефиниции на класове вътре в други класове. Позволява се логическото им групиране и контролиране на видимостта.

**Статични вложени класове** - Не е необходимо съдържащият клас на вътрешни класове да бъде инстанциран, за да се използват

Дефинира се с ключовата дума "abstract". Всеки клас, който има поне един абстрактен метод, трябва да бъде деклариран като абстрактен.

* **Абстрактен метод** - Mетод без тяло, само с декларация. Дефинира се с ключовата дума "abstract".

**Чист абстрактен клас** - Съдържа в себе си само абстрактни методи. Не може да съдържа член-променливи

**Виртуален метод** - Метод, който може да бъде пренаписан (override). По подразбиране всички  методи в Java са виртуални.

**Интерфейс**- Дефинира поведение без изрична имплементация. Един клас може да имплементира много интерфейси (използва се ключова дума "implements"). В интерфейсите може да има само абстрактни методи и константи.

* Класът **ArrayList** - Динамично разширяем масив. Важни методи: get(), add(), remove(), clear().
* **Принципи на ООП** (абстракция, капсулация, наследяване, полиморфизъм);
* **Абстракция:**игнорираме детайлите и виждаме само това което ни интересува**;**
* **Капсулация:**скриване на детайлите на класовете, посредством модификатори за достъп;

**Наследяване**на класове: Не е възможно множествено наследяване с класове в Java (един клас може да има само един клас-родител - single inheritance);

**Полиморфизъм:**дефинира как се работи по еднакъв начин с различни обекти, които дефинират специфична имплементация на абстрактно поведение

* **Модификатори за достъп**: public, private, protected, package, default.
* **Статичните**(static) полета и методи са свързани с класовете, а не с конкретен обект от даден клас. Използват се без да има създадена инстанция/обект  на класа.
* **Нестатичните**(non-static) полета и методи са свързани с обект от даден клас. Следователно, нестатичните членове на клас се извикват с помощта на създаването на обект. По подразбиране всички променливи и методи са non-static.
* **Пакети**(packages) в Java - пакетите позволяват на разработчиците да групират своите класове на база определени критерии.