

2

.1

خلاصه تقاضه Shallow Copy و Deep Copy :

کپی سطحی - Shallow Copy

- فقط اشاره‌گرها (رفرانس‌ها) را کپی می‌کند.

- تغییر در داده‌های تو در تو روی نسخه اصلی هم اثر می‌گذارد.

- سریع‌تر و کم‌صرف‌تر از نظر حافظه.

- مناسب برای داده‌های ساده و غیرتو در تو.

سرعت و مصرف حافظه مهم است و داده‌ها ساده هستند Shallow Copy

سناریو : تمام فیلد‌ها immutable یا primitive باشند - در حافظه صرف جویی می‌کنند

کپی عمیق - Deep Copy

- یک کپی کاملاً مستقل از شیء و تمام اجزای تو در تو آن می‌سازد.

- تغییرات در کپی، روی نسخه اصلی تأثیری ندارد.

- کندتر و پرمصرف‌تر از نظر حافظه.

- مناسب برای داده‌های پیچیده و تو در تو.

تغییرات روی کپی، اصل داده را تغییر نمی‌دهد Deep Copy

سناریو : شی دارای فیلد mutable باشد - وقتی میخواهیم نسخه اصلی و کپی کاملاً جدا سازی شوند.

برای متدهای shallow copy , Copy مناسب.

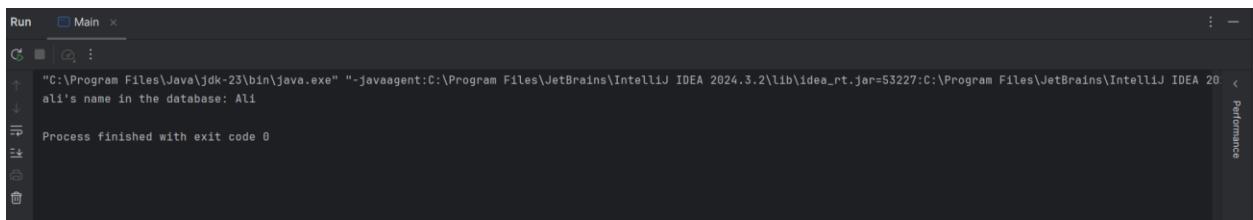
چون primitive است که string , name int , id است و primitive immutable است.

.2

میتوان هنگام Override کردن یک متود نوع بازگشتی ان را به یک زیرنوع تغییر داد که به این ویژگی Covariant return types میگویند.

یک زیرنوع از Entity است. و تغییر نوع بازگشتی از Entity به Human مجازه چون هر Human یک Entity است.

.3



.4

متدهای clone()

متدهای clone() در برنامهنویسی برای ایجاد یک کپی از شیء موجود استفاده میشود. این متدهای از اینترفیس Cloneable ارثبری میکند و به شیء اجازه میدهد خود را کپی کند.

عملکرد clone()

پیشفرض یک Shallow Copy (کپی سطحی) ایجاد میکند.

یعنی فقط فیلدهای اصلی شیء کپی میشوند و اگر شیء شامل اشیا تو در تو باشد، فقط رفرنس آنها کپی میشود.

برای Deep Copy باید متدهای clone() را اورراید کنید و کپی عمیق را دستی پیادهسازی کنید.

کاربردهای clone()

1. ایجاد کپی از اشیا بدون وابستگی به نسخه اصلی مثلاً در الگوهای طراحی مثل Prototype).

.5

3. بهینه‌سازی عملکرد با کاهش نیاز به ساخت اشیا جدید از صفر.

2. جلوگیری از تغییرات ناخواسته روی داده‌های اصلی.

Cloneable یک مارکر اینترفیس است و هیچ متدهای ندارد.

فقط اجازه فراخوانی `clone()` را می‌دهد (در غیر این صورت خطای خطای می‌دهد).

طراحی آن قدیمی است و امروزه معمولاً از کپی سازنده یا کتابخانه‌های کمکی استفاده می‌شود.