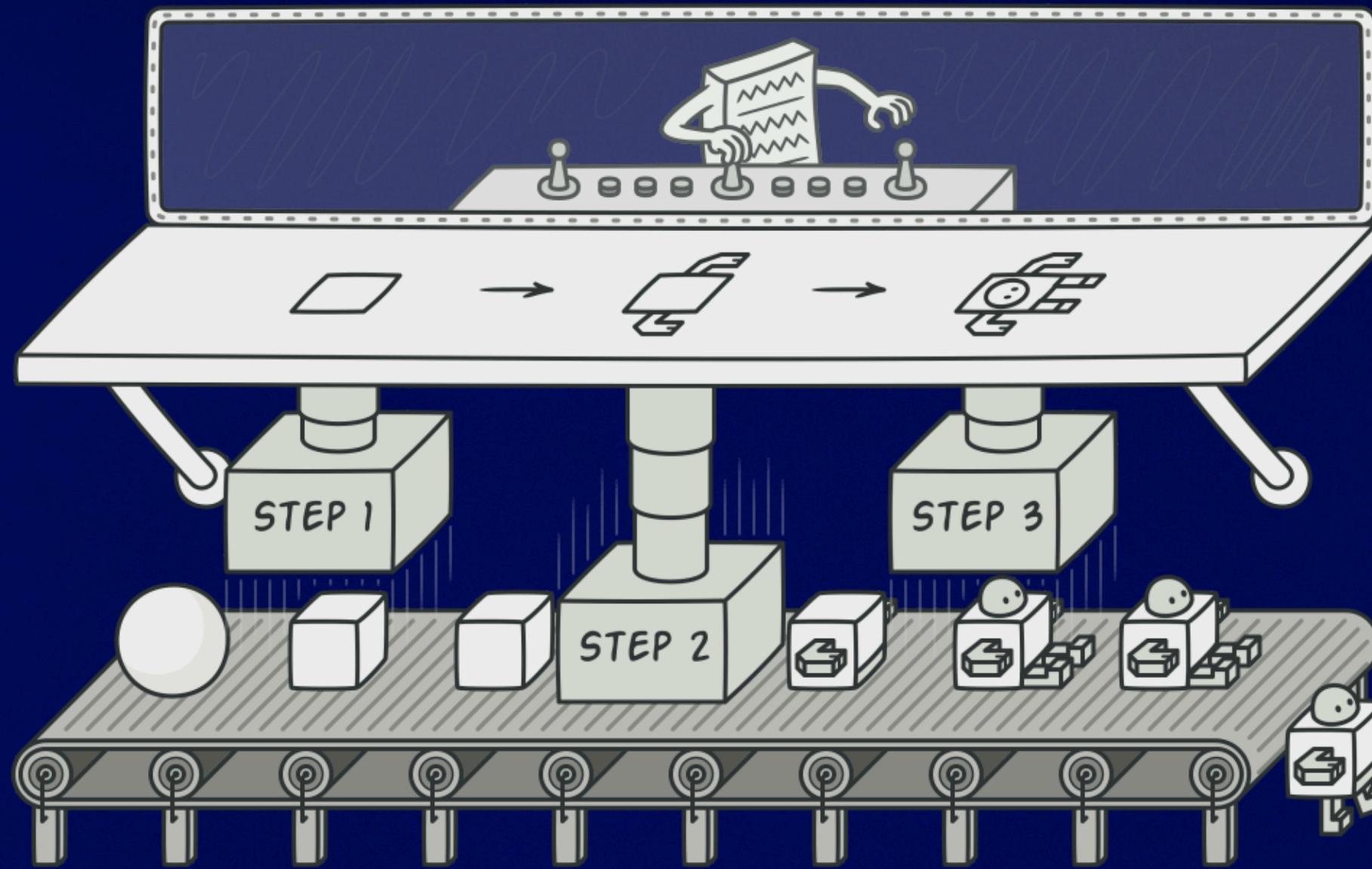


الگو سازنده یا (BUILDER PATTERNS)



معرفی

BUILDER PATTERN ◆

- یک CREATIONAL PATTERN برای ساخت اشیاء پیچیده.
- هدف: جدا کردن کد ساخت (CONSTRUCTION) از کلاس اصلی و مدیریت آن در اشیاء جداگانه.
- مناسب زمانی که ساخت یک شئ شامل چند مرحله و پارامتر است.

مشکل

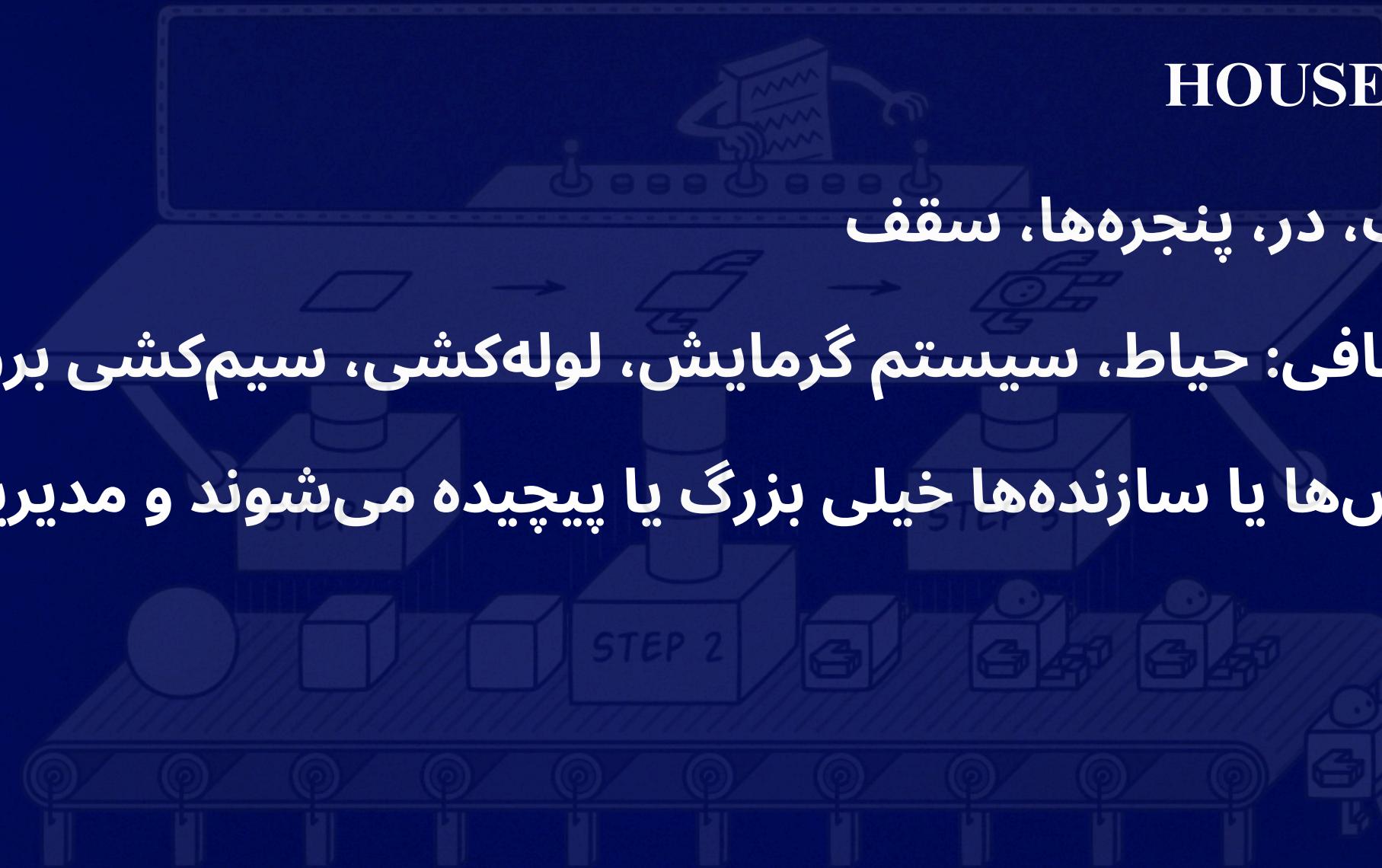
✓ اشیاء پیچیده نیاز به ساخت مرحله به مرحله دارند و شامل چندین فیلد و اشیاء تو در تو هستند.

✓ کد ساخت اغلب در سازنده‌های بزرگ با پارامترهای زیاد قرار من‌گیرد یا در کل کد مشتری پراکنده است.

مثال

♦ ساخت یک HOUSE

- دیوارها، کف، در، پنجره‌ها، سقف
 - امکانات اضافی: حیاط، سیستم گرمایش، لوله‌کشی، سیم‌کشی برق
- ✖ مشکل: کلاس‌ها یا سازنده‌ها خیلی بزرگ یا پیچیده منشوند و مدیریت آن سخت است.



راه حل های اشتباه

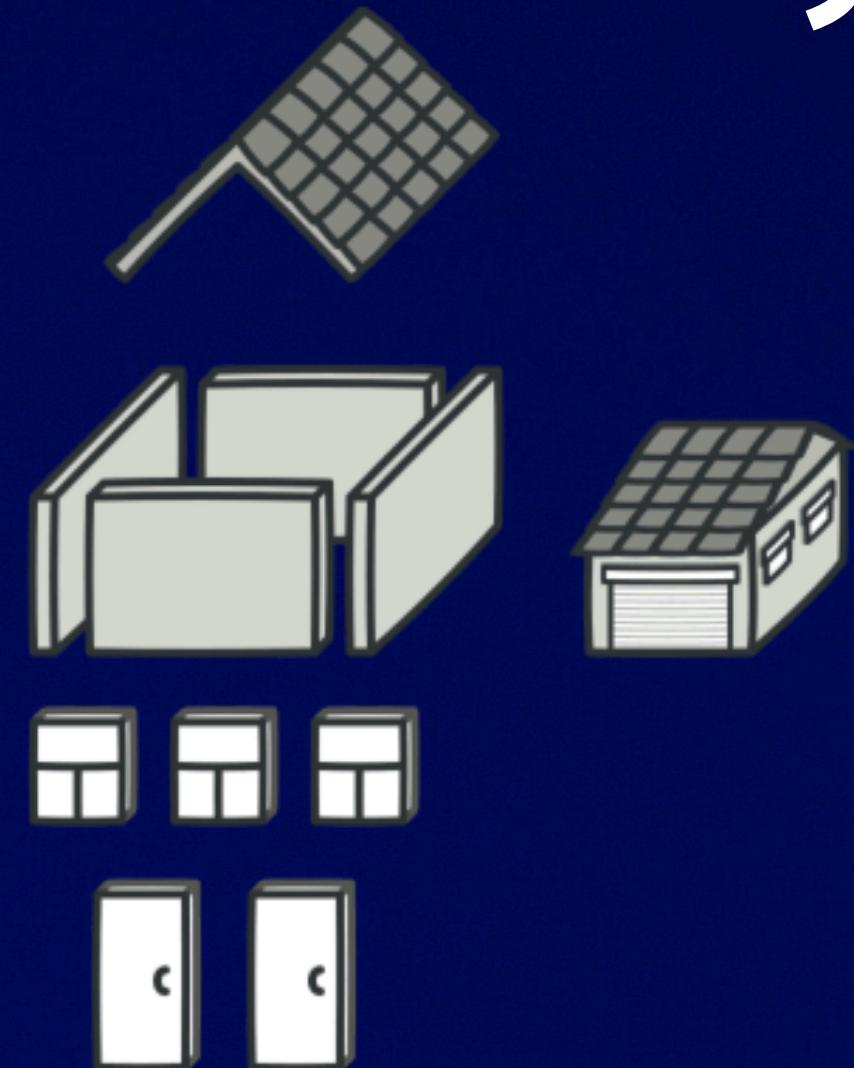
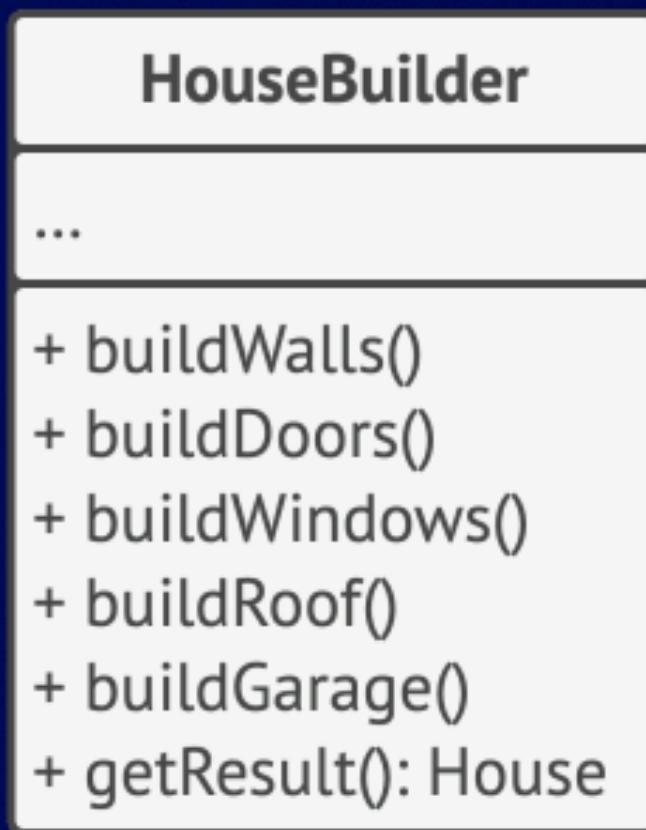
- ایجاد زیرکلاس های مختلف برای همه ترکیب های پارامترها → منجر به تعداد زیادی کلاس می شود.
- ایجاد یک سازنده عظیم با تمام پارامترها → اغلب پارامترها استفاده نمی شوند و کد زشت می شود.

راه حل درست

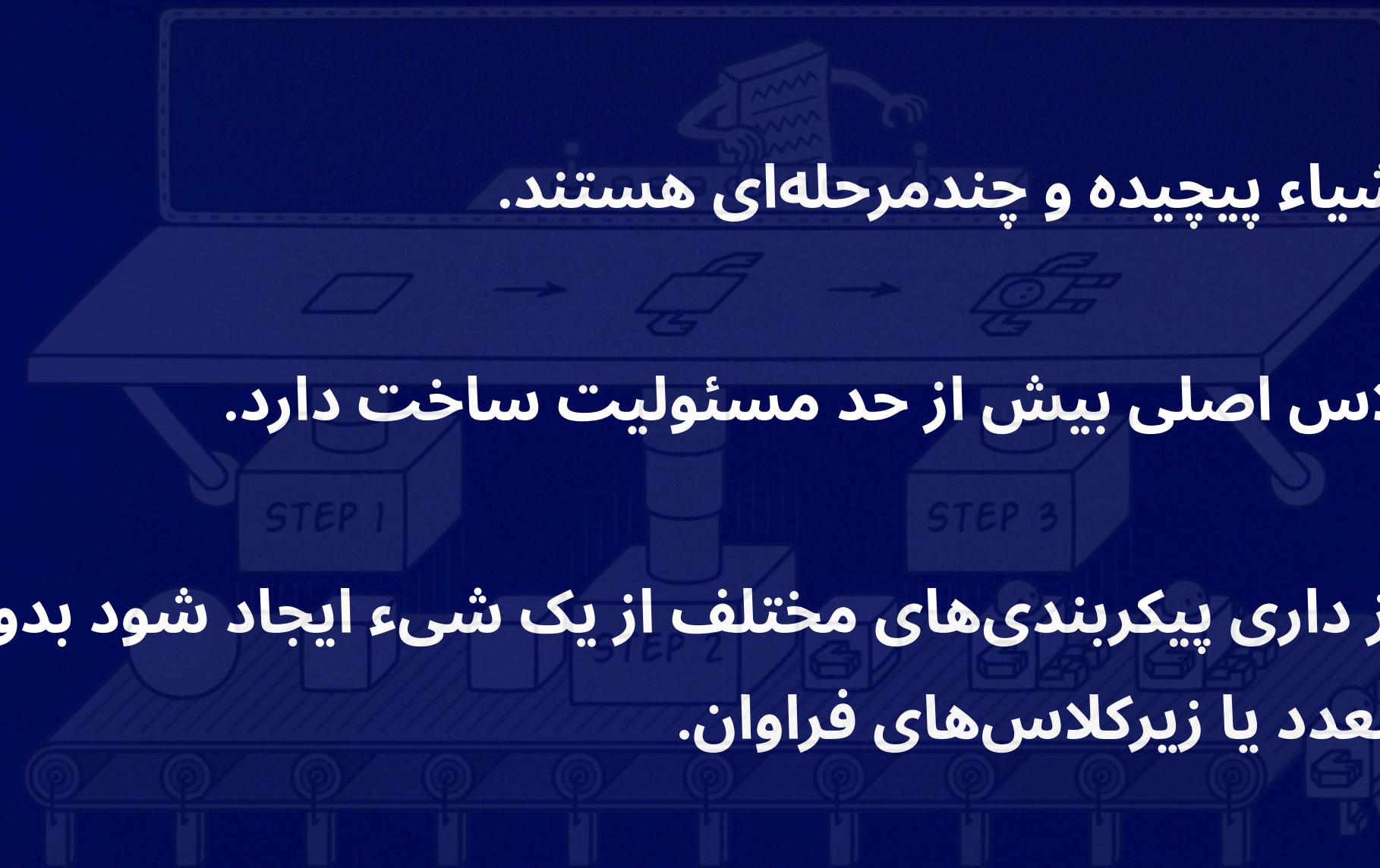
→ استخراج کد ساخت به اشیاء جدآگاه (BUILDERS) → BUILDER PATTERN

- ساخت اشیاء به صورت مراحل مشخص (مثلًاً، BUILDWALLS (BUILDDOOR
- فقط مراحل لازم برای پیکربندی خاص شئ فرآخوانی می‌شوند

راه حل درست



کی استفاده کنیم



✓ وقتی اشیاء پیچیده و چندمرحله‌ای هستند.

✓ وقتی کلاس اصلی بیش از حد مسئولیت ساخت دارد.

✓ وقتی نیاز داری پیکربندی‌های مختلف از یک شیء ایجاد شود بدون ایجاد سازنده‌های متعدد یا زیرکلاس‌های فراوان.

رابطه با الگو FACTORY

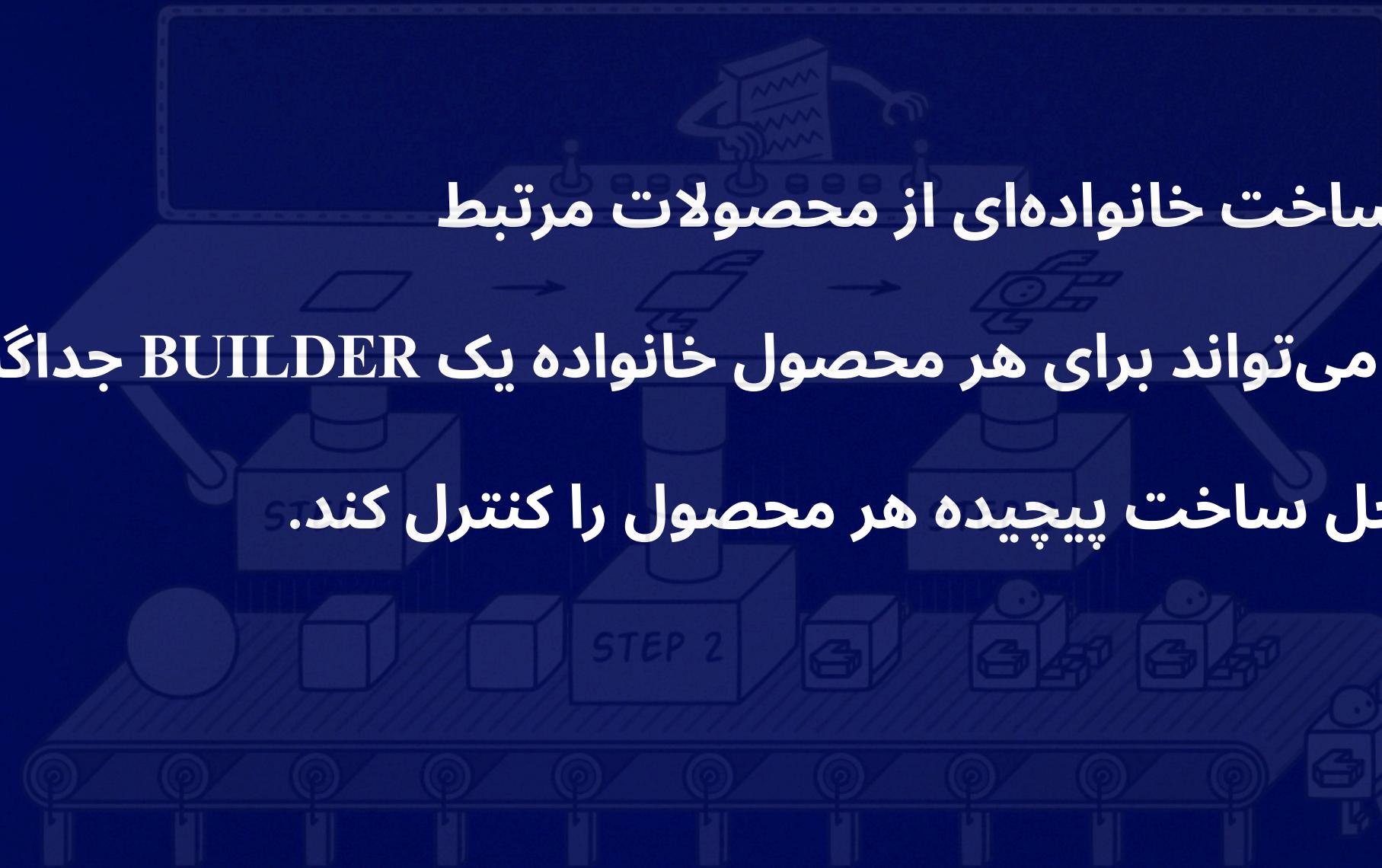
تمركز روی ساخت یک محصول منفرد FACTORY PATTERNS ◆

می تواند از FACTORY METHOD برای ایجاد قطعات استفاده BUILDER

کند، اما کنترل مراحل ساخت را در اختیار دارد.

رابطه با الگو ABSTRACT FACTORY

- ♦ مدیریت ساخت خانواده‌ای از محصولات مرتبط
- ♦ می‌تواند برای هر محصول خانواده یک BUILDER جداگانه داشته باشد تا مراحل ساخت پیچیده هر محصول را کنترل کند.



BUILDER VS FACTORY

→ تمرکز روی نوع و کلاس محصول ◆

→ تمرکز روی مراحل و ترتیب ساخت و پیگریندی انعطاف‌پذیر ◆

نکته:  ABSTRACT FACTORY و BUILDER می‌توانند با هم ترکیب شوند تا خانواده‌ای از محصولات پیچیده با مراحل ساخت انعطاف‌پذیر ایجاد کنند.

نکات کلیدی:

- بک اشیاء جداگانه برای ساخت مرحل مختلف. → BUILDER ✓
- .BUILDER (اختیاری) → مسئول اجرای مرحل بر روی DIRECTOR ✓
- . BUILDWALLS(), BUILDDOOR(), ... → BUILDR ✓ متدهای
- مزیت: پیکربندی انعطاف‌پذیر و خوانایی بالا ✓

جمع بندی

- ★ **BUILDER** جدا از کلاس اصلی، کد ساخت اشیاء پیچیده را مدیریت می‌کند.
- ★ هر مرحله ساخت به صورت متد مستقل در **BUILDER** قرار دارد.
- ★ امکان ساخت پیکربندی‌های متفاوت بدون تغییر در کلاس اصلی وجود دارد.
- ★ مناسب برای اشیاء با پارامترهای متعدد یا تو در تو.