OOA & OOD

البرمجة مش مجرد كتابة كود، هي في الأساس طريقة تفكير وتنظيم
تخيل لو عندك مشروع ضخم زي نظام لمستشفى أو بنك، هل ينفع تبدأ تكتب الكود عشوائي؟ أكيد لأ، لأنك هتضيع وسط التفاصيل.
عشان كده ظهر مفهوم OOA و OOD
الفكرة إنك قبل ما تمسك الكيبورد وتكتب أي سطر كود، بتقعد تحلل الأول: إيه المطلوب؟ مين ال Objects اللي موجودة؟ وإزاي هيتعاملوا مع بعض؟ وبعد كده تبدأ تصمم الكود بشكل منظم يخلي البرنامج سهل يتطور ويتعدل من غير ما تبني كل حاجة من جديد.

Object Oriented Analysis and Design (OOAD)

هو طريقة بنصمم بيها البرامج عن طريق إننا نفكر في كل حاجة كأنها Object زي الحاجات اللي في الحياة الواقعية.
إحنا الأول بنفهم النظام محتاج يعمل إيه، وبعدها نحدد ال Objects الأساسية اللي موجودة فيه، وفي الآخر بنقرر الObjects دي هتتعاون مع بعض إزاي. الطريقة دي بتخلي البرامج أسهل في الإدارة، وسهلة نعيد استخدامها، وكمان تقدر تكبر وتتطور من غير ما نعقد الدنيا

اهم مكونات OOAD Object-Oriented Programming (OOP)

بيمثل الحاجات الواقعية ك Objects جوه البرنامج.
Design Patterns: حلول جاهزة لمشاكل متكررة في التصميم، بتسهل على المطورين.
UML Diagrams: رسومات بتوضح مكونات النظام والعلاقات بينهم.
Use Cases: سيناريوهات بتوضح إزاي المستخدم هيتعامل مع النظام، وبتساعد نفهم المتطلبات.

Object-Oriented Analysis (OOA)

هو إننا نفهم المتطلبات عن طريق إننا نبص للمشكلة كأنها Objects بتمثل حاجات أو مفاهيم من الواقع ليها علاقة بالنظام. بنحدد خصائصها (Attributes)، وظائفها (Behaviors)، والعلاقات بينهم.
مش بنفكر في التنفيذ هنا، بس في "إيه اللي مطلوب؟".

مثال من لعبة:

الشخصيات = Objects.
لكل شخصية صفات (الطاقة، السرعة) ووظائف (يجري، يقفز).
نعمل زي "خريطة" تبين كل الحاجات اللي اللعبة محتاجاها.
OOA كأنه بنكتب to-do list للبرنامج: يتحرك اللاعب، يسجل النقاط، يحفظ المستوى... إلخ.

Object-Oriented Design (OOD)

بعد ما نعمل التحليل، بنبدأ في التصميم: نحول النموذج اللي عملناه في OOA إلى تصميم تفصيلي.
نحدد بالظبط ال Objects دي هتخزن بيانات إزاي (Data Organization). نشرح الوظائف خطوة بخطوة (Procedural Description).
يعني هنا خلاص بنجهز الخطة اللي هنمشي عليها لما نكتب الكود.

Benefits of Object-Oriented Analysis and Design (OOAD)

- سهولة الصيانة والتوسع: OOAD بيشجع على تقسيم النظام لأجزاء صغيرة (Modules) قابلة لإعادة الاستخدام، وده بيخلي التعديل والتوسع أسهل مع الوقت.
- وضوح النظام: بيوفر تمثيل مجرد (Abstract) للنظام، يخلي فهمه أسهل سواء للمطور أو أي حد جديد على المشروع.
- إعادة الاستخدام: المبادئ الخاصة بالـ OOD بتخلي في إمكانية لإعادة استخدام الـ Objects بدل ما نكتب نفس الكود أكتر من مرة، وده بيرفع من جودة البرنامج.
- التعاون بين الفريق: OOAD بيوفر لغة موحدة وأسلوب واحد يخلي التواصل بين المبرمجين في الفريق أوضح وأسهل.
- قابلية التوسع: بيساعد على تصميم أنظمة برمجية قابلة للتطوير مع تغير احتياجات المستخدم أو متطلبات السوق.

Challenges of Object-Oriented Analysis and Design (OOAD)

- زيادة التعقيد: وصف الـ Objects والتعامل مع تفاعلاتها ممكن يزود تعقيد النظام.
- أداء أقل شوية: لإنشاء وإدارة الـ Objects ممكن يسبب شوية حمل إضافي (Overhead) على النظام.
- صعوبة التعلم: المبتدئين ممكن يلاقوا الموضوع صعب في البداية لأنه محتاج فهم كويس لمبادئ الـ OOP.
- وقت وتكلفة: OOAD محتاج تخطيط مسبق ووثائق (Documentation) كتير، وده ممكن يزود مدة التطوير والتكلفة.
- أغلى شوية من غيره: بسبب التخطيط والـ Documentation الكتيرة، ممكن يكون أغلى من منهجيات تانية.

Real-World Applications of OOAD

- الأنظمة البنكية: زي ما في البنوك بيتعاملوا مع معاملات مالية معقدة، عملاء، حسابات ... OOAD بيساعد في تنظيم ده بشكل مرن وقابل للتوسع.
- الملفات الطبية الإلكترونية (EHR): بيساعد في تنظيم بيانات المرضى والسجلات الطبية بطريقة مرنة وسهلة التطوير مع ظهور متطلبات جديدة.
- أنظمة التحكم في الطيران: بيساعد على تمثيل التفاعل بين أنظمة الملاحة، الحساسات، وأساليب التحكم عشان نضمن الأمان والموثوقية.
- أنظمة الفوترة في الاتصالات: OOAD بيساعد في تصميم أنظمة بتتعامل مع خطط اشتراك معقدة وفواتير وبيانات العملاء بشكل منظم وسهل التوسع.
- منصات التسوق الإلكتروني: زي أمازون مثلًا، بيتم استخدام OOAD لتنظيم الكاتالوج، بيانات المستخدمين، عربيات التسوق، وعمليات الدفع.

المصادر:

https://www.geeksforgeeks.org/software-engineering/object-oriented-analysis-and-design/