Minggu pertama ini tidak terlalu banyak ilmu baru, menurut saya minggu ini seperti mengingat kembali pelajaran yang pernah diajarkan sewaktu saya masih kuliah, hanya saja menggunakan bahasa Ruby. Tidak terasa membosankan walaupun saya harus mengulang pelajaran yang pernah saya pelajari, karna instruktur banyak memberikan studi kasus dan berusaha mengajak para peserta untuk terus ikut aktif dalam memecahkan sebuah masalah.

Pada pertemuan pertama saya belajar mengenai perbedaan prosuderal programming dan OOP. Keduanya memiliki keunggulan masing-masing dan memiliki kegunaan yang sama, sama-sama bisa memecahkan masalah pemrograman, namun disini saya belajar bahwa prosedural programming berorientasi pada aksi sedangkan OOP berorientasi pada objek.

Sebelum saya mempelajari tentang OOP di pertemuan minggu pertama program Generasi GIGIH tingkat intermediate saya hanya memahami konsep OOP dan bagaimana cara OOP bekerja, namun sekarang saya mendapat sudut pandang baru bagaimana OOP dapat memecahkan permasalahan baik besar maupun kecil dengan cara membuat beberapa entity dari satu kata kunci utama. Contohnya seperti studi kasus yang telah dipelajari minggu kemarin yaitu tentang game. Langkah pertama yang harus saya lakukan adalah mencari class apa saja yang akan saya butuhkan untuk memenuhi sebuah kriteria game. Nama utama dari class ini haruslah berupa kata kunci dari tema permasalahan itu sendiri, yaitu ‘Game’.

Pada minggu ini saya juga mempelajari tentang inheritance dan polymorphism pada OOP. Sebenarnya dulu saya sudah mendapatkan materi ini pada saat di perkuliahan. Disini saya seperti mengingat kembali tentang inheritance yang mana dulu saya belajar inheritance dari pemrograman PHP dan Java. Saya menyadari bahwa setiap bahasa memiliki beberapa perbedaan dalam menerapkan OOP. Java yang terkenal karna kerumitannya yang hanya untuk menuliskan 1 buat class beserta komponennya. PHP yang hampir mirip Java tetapi lebih dipermudah. Karna di PHP ini saya menemukan mekanisme Trait, konsepnya mirip dengan inheritance pada Ruby, hanya saja Trait memungkinkan sebuah class memiliki parent lebih dari satu.

Hal yang masih belum saya pahami adalah kapan waktu yang tepat untuk menggunakan konsep composition. Apakah composition hanya digunakan pada kelas utama yang memerlukan banyak kelas dan sehingga perlu menggunakan composition daripada inheritance? Apakah tidak mungkin menggunakan konsep composition pada class dependent? Sepertinya saya harus mengerjakan studi kasus yang berbeda untuk menemukan jawaban dari pertanyaan tersebut.