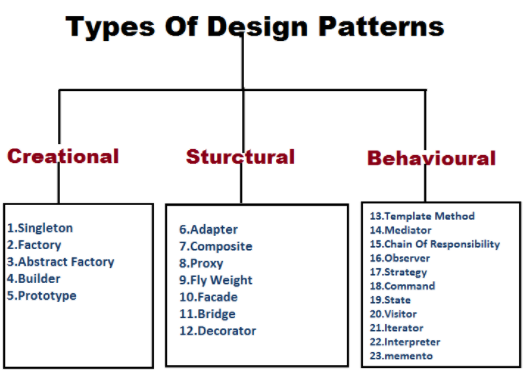
**Overcome reflection issue:** To overcome issue raised by reflection, [enums](https://www.geeksforgeeks.org/enum-in-java/) are used because java ensures internally that enum value is instantiated only once. Since java Enums are globally accessible, they can be used for singletons. Its only drawback is that it is not flexible i.e it does not allow lazy initialization.

As enums don’t have any constructor so it is not possible for Reflection to utilize it. Enums have their by-default constructor, we can’t invoke them by ourself.**JVM handles the creation and invocation of enum constructors internally.** As enums don’t give their constructor definition to the program, it is not possible for us to access them by Reflection also. Hence, reflection can’t break singleton property in case of enums.



1. **Creational**: These patterns are designed for class instantiation. They can be either class-creation patterns **or object-creational patterns**.

2. **Structural**: These patterns are designed with regard to a class's structure and composition. The main goal of most of these patterns is to increase the functionality of the class(es) involved, without changing much of its composition.

3. **Behavioral**: These patterns are designed depending on how one class **communicates with others**.

https://www.freecodecamp.org/news/the-basic-design-patterns-all-developers-need-to-know/

Object class defines the equals and hascode method

**The default implementation of equals() in the class Object says that equality is the same as object identity. And income and expenses are two distinct instances.**