# Overview:

* A program should not depend on how objects are created and arranged.
* Instantiation is an activity that should not always be done in public and can often lead to coupling problems.
* In Java, the simplest way to create an instance of an object is by using the new operator.
  + fred = new Fred(); // Instance of Fred class.
  + Creates a concrete class.
* Tying your code to a concrete class can make your code more fragile & less flexible.
  + Code may have to be changed as new concrete classes are added.
  + Your code will not be "closed for modification" (violation of the Open/Closed Principle).
    - To extend it with new concrete types, you will have to reopen it.
* Creational design patterns provide a way to create objects.
* Creational design patterns abstract the instantiation process :
  + The creation logic is hidden.
  + Encapsulates knowledge about which concrete classes the system uses.
  + Programmer may call a method or use another object, rather than instantiating objects directly using the new operator.
* All the system at large knows about the objects is their interfaces as defined by abstract classes.
  + Gives the programmer a lot of flexibility in what gets created, who creates it, how it gets created, and when.
  + Let you configure a system with "product" objects that vary widely in structure and functionality.
  + Configuration can be static (compile-time) or dynamic (at run-time).
* By coding to an interface, you can insulate yourself from a lot of changes that might happen to a system down the road.
* Sometimes creational patterns are competitors :
  + There are cases when either Prototype or Abstract Factory could be used profitably.
* Sometimes creational patterns are complementary :
  + Builder can use one of the other patterns to implement which components get build.
  + Prototype can use Singleton in its implementation.
* Class Patterns vs Object Patterns (sub-categories) :
  + Class patterns describe how relationships between classes are defined :
    - Use inheritance.
    - Relationships are established at compile time.
    - Factory pattern.
    - Drawback of this approach is that it can require creating a new subclass just to change the class of the product.
    - Changes can cascade.
    - When the product creator is itself created by a factory method, then you have to override its creator as well.
  + Object patterns describe relationships between objects :
    - Use composition.
    - Relationships are typically created at runtime.
      * More dynamic and flexible.
    - Abstract factory, singleton, builder, & prototype patterns.
  + There are five creational patterns that we will study :
    - We will highlight their similarities & differences.
    - (Factory, Abstract Factory, Singleton, Builder, Prototype).