YOU NEED TO KNOW

PATTERNS!
COMMAND PATTERN ENCAPSULATES A REQUEST AS AN OBJECT, THEREBY LETTING USERS PARAMETERIZE CLIENTS WITH DIFFERENT REQUESTS, QUEUE OR LOG REQUESTS, AND SUPPORT UNDOABLE OPERATIONS.
FLYWEIGHT PATTERN USES SHARING TO SUPPORT LARGE NUMBERS OF FINE-GRAINED OBJECTS EFFICIENTLY.
**STATE** PATTERN ALLOWS AN OBJECT TO ALTER ITS BEHAVIOR WHEN ITS INTERNAL STATE CHANGES. THE OBJECT WILL APPEAR TO CHANGE ITS CLASS.
STRATEGY PATTERN DEFINES A FAMILY OF ALGORITHMS, ENCAPSULATE EACH ONE, AND MAKES THEM INTERCHANGEABLE. STRATEGY LETS THE ALGORITHM VARY INDEPENDENTLY FROM CLIENTS THAT USE IT.
TYPE OBJECT PATTERN ALLOWS THE FLEXIBLE CREATION OF NEW "CLASSES" BY CREATING A SINGLE CLASS, EACH INSTANCE OF WHICH REPRESENTS A DIFFERENT TYPE OF OBJECT.
**Observer** pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.
**Singleton** pattern ensures a class only has one instance, and provides a global point of access to it.
SERVICE LOCATOR PATTERN PROVIDES A GLOBAL POINT OF ACCESS TO A SERVICE WITHOUT COUPLING USERS TO THE CONCRETE CLASS THAT IMPLEMENTS IT.
PROTOTYPE PATTERN SPECIFIES THE KINDS OF OBJECTS TO CREATE USING A PROTOTYPICAL INSTANCE, AND CREATE NEW OBJECTS BY COPYING THIS PROTOTYPE.
**Abstract Factory** pattern provides an interface for creating families of related or dependent objects without specifying their concrete classes.
OBJECT POOL PATTERN IMPROVES PERFORMANCE AND MEMORY USE BY REUSING OBJECTS FROM A FIXED POOL INSTEAD OF ALLOCATING AND FREEING THEM INDIVIDUALLY.
**Builder** pattern separates the construction of a complex object from its representation so that the same construction process can create different representations.
OPEN BOOK
READ ONLY MATERIAL ALLOWED
LEARN FOR THE EXAM
YOU DO NOT HAVE TIME TO GOOGLE FOR EVERYTHING