DDR Checklist – Mohid Nadeem

#	Design Patterns	Week
1	Façade	3b, 3c
2	Observer	4b
3	Adaptor	4c
4	Singleton	4c
5	~~Factory~~	4c
6	Decorator	5b
7	Abstract Factory	5c
8	Strategy / Policy	5d
9	Command	8a
10	~~Template~~	8b
11	Iterator	8b
12	Composite	9a
13	State	9b
14	Proxy	10a
15	Chain of Responsibility	10b
16	Memento	15a
17	Dependency Injection	16a

#	Refactoring	Opposites
1	Add Parameter	
2	Change Association (Bi to Uni – Directional)	Change Association (Uni to Bi – Directional)
3	Reference to value	Value to reference
4	Extract subclass Extract superclass	Collapse hierarchy
5	Consolidate conditionals	
6	Consolidate duplicate conditionals	
7	Decompose conditional	
8	Encapsulate collection	
9	Encapsulate downcast	
10	Encapsulate field	
11	Extract class Extract Interface	Inline Class
12	Extract method	Inline method
13	Extract variable	
14	Form template method	
15	Hide delegate	Remove Middle Man
16	Hide method	
17	Inline temp	
18	Replace Temp with Query	
19	Remove Assignments to Parameters	
20	Introduce Parameter Object	
21	Preserve Whole Object	

#	Bad Smell in Code	Description
1	Duplicated Code	Same code structure appearing multiple times in different places.
2	Long Method	Methods that are too long and try to do too much.
3	Large Class	Classes that have grown too large and encompass too many responsibilities.
4	Long Parameter List	Methods that have too many parameters, making them hard to understand and use.
5	Divergent Change	A class that is commonly changed in different ways for different reasons.
6	Shotgun Surgery	Making a single change requires altering many different classes.
7	Feature Envy	A method that seems more interested in a class other than the one it is in.
8	Data Clumps	Groups of data that frequently appear together and should be encapsulated.
9	Primitive Obsession	Overuse of primitive data types instead of small objects for simple tasks.
10	Switch Statements	Complex switch or case statements that are often better handled with polymorphism.
11	Parallel Interface Hierarchies	Similar hierarchies that are parallel to each other and require parallel changes.
12	Lazy Class	Classes that have no real purpose and do not contribute enough to justify their existence.
13	Speculative Generality	Code that is designed to be reusable or flexible in ways that are not currently needed.
14	Temporary Field	Fields that are only sometimes needed, leading to code complexity.
15	Message Chains	Long chains of method calls to get some data, leading to fragile code.
16	Middle Man	A class that does too little, delegating almost everything to another class.
17	Inappropriate Intimacy	Classes that know too much about each other's internal details.
18	Incomplete Library Class	Using a library class but needing to add extra functionality that isn't provided.
19	Data Class	Classes that have fields but little or no methods to operate on them.
20	Refused Bequest	Subclasses that do not want or need everything they inherit from their parent classes.