מרכז ההדרכה המוביל למקצועות היי-טק
המרכז הישראלי לטכנולוגיה ותקשורת

IITC InterBit

GoF Design Patterns:
Elements of Reusable Object-Oriented Software
Training Program Schedule

• Time	• Day 1	• Day 2	• Day 3	• Day 4	• Day 5
09:00 – 09:30	• Introduction • GoF Design Patterns Elements of reusable object oriented software: Book Introduction • Academic Discussion	• Summary • Cloning an object in Java – cont' • shallow clone vs. deep clone	• Summary • Bridge – separate abstraction from implementation so the two can vary independently	• Summary • Structural Design Patterns – Evaluation Exam Preparation Quiz	• Summary
09:30 – 10:00	• GoF Design Patterns Academic Discussion – Cont.	• Code examples • Cloning with serialization / de-serialization	• The Java Collections Framework – Code Example	• Exam Preparation Quiz – Cont.	• State – class delegates action to internal State variable so it behaves differently at different states
10:00 – 10:30	• Env. + IDE setup & configurations • Importing training materials to IDE	• Code examples • prototype lab exercise • Exercise review	• Bridge lab exercise • Exercise review	• **Module 4:** Behavioral Design Patterns	• Code Examples • State lab exercise • Exercise review
10:30 – 11:00	• **Module 1:** Design Patterns – Motivation • Basic Design Concepts • separate abstract requirements – into an interface • isolate the minimum factor of change – D.R.Y • favor composition over inheritance • Code Examples: File Utils	• Singleton – A class of which there can be only one instance • Lazy vs. Eager Initialization • Eager and easy Singletons • Enum singletons • Utility class singletons • Thread safe initialization	• Composite – an object may consist of other objects recursively • Recursive composition • Code Examples • Combining design patterns: other patterns use composite too	• Chain of Responsibility – pass a request through a chain of objects until it encounters the most appropriate handler	• Template – abstract definition of an algorithm • Code Examples
11:00 – 11:30	• Design Patterns Taxonomy • Creational Patterns – strategies for creating new objects • Structural Patterns – how objects may be grouped into more complex structures • Behavioral Patterns – define the flow of communication between objects	• synchronized instantiation method • using static instances as singletons • using the volatile key word with the **d**ouble **c**hecked **l**ock pattern	• Composite lab exercise • Exercise review	• Code Examples • Class Loaders in Java – reversed chain of responsibilities	• Template lab exercise • Exercise review
11:30 – 12:00	• **Module 2:** Creational Design Patterns • Factory (Method) – a central point for instance creation	• Code Examples	• Decorator – dynamically add functionality to objects • Code Examples	• Command – execute pieces of code, oblivious to implementation • Code Examples	• Visitor – allow a visitor class to perform operations on a visited class
12:30 – 13:30	---------- Lunch break ----------				

רח' החילזון רמת גן מיקוד - טלפון: - פקס
Http://www.iitc.co.il

13:30 – 14:00	• Factory (Method) – a central point for instance creation – cont - caching, pooling, discussion • Code examples	• Singleton lab exercise • Exercise review	• Decorator lab exercise • Exercise review	• Command lab exercise • Exercise review	• Variations of the Visitor design pattern • Code Examples
14:00 – 14:30	• Abstract Factory – select between several possible factories where each factory generates it's own family of objects • Code Examples	• singleton and serialization in java • defining a readResolve() method • Code Examples	• Proxy – access to objects can be controlled by another object with the same interfaces • The proxy UML class diagram simplicity – what we don't see	• Iterator – traverse a collection of data within a class • Code Examples	• Interpreter - parse a language expression into a matching tree • Code Examples – Roman Numerals
14:30 – 15:00	• How to complete the exercises • Abstract factory lab exercise	• Creational Design Patterns – summary • Creational Design Patterns – Evaluation Exam Preparation Quiz	• Importance of the proxy design pattern • What it takes to implement a Proxy • Code Examples	• Iterator lab exercise • Exercise review	• Conclusion
15:00 – 15:30	• Exercise review • Builder – separate the construction algorithm from the internal representation to construct complex objects with multiple parts	• **Module 3:** Structural Design Patterns • Overview of structural design patterns • Composition vs. Inheritance • Composition and Delegation • Aggregation vs. Association	• Dynamic Proxy – the Invocation handler interface and the Proxy factory in Java • Implementing a dynamic proxy with reflection API • Code Examples	• Observer – a way for an object to notify others when it change • The Observable / Observer classes in Java • Code Examples	• Behavioral Design Patterns – Evaluation Exam Preparation Quiz
15:30 – 16:00	• Code Examples • Builder lab exercise	• Adapter – force a class to conform to a new interfaces	• **Mid Module Summary**: Do all these patterns look the same? • Structural Design Patterns analysis – wrapper/delegate structure and inheritance	• Mediator – classes communicate through a mediator for simplification and loose coupling • Code Examples	• General - Evaluation Exam Preparation Quiz
16:00 – 16:30	• Exercise review • Prototype – clone an existing object-creation	• Class Adapter • Object Adapter	• Facade – simplify the access to complex systems • The DAO pattern – simplify data access • Code Examples	• Memento – capture and externalize an object's state so it can be restored later • Tag / Flag Interfaces • Code Examples	• **Evaluation Exam**
16:30 – 17:00	• Cloning an object in Java	• Code Examples	• Flyweight – remove state variables and replace with parameters to share objects in order to save allocations • Code Examples	• Strategy – encapsulate an algorithm in a class • Code Examples	

רח' החילזון רמת גן מיקוד - טלפון: - פקס
Http://www.iitc.co.il