**Lab # 8**

**SOFTWARE DESIGN**

**AND**

**THE SOFTWARE DESIGN DOCUMENT (SDD)**

**Objectives:**

* Deeper understanding of software design and the software design document (SDD)
* Learn how to find the relationships between classes to create UML class diagram

1. **Outline**

* Software design concepts and principals
* Software architecture
* Specifying the attributes and the operations and finding the relationships between classes
* Creating UML class diagram
* Software design document.

1. **Background**

The purpose of software design is “to produce a workable (implementable) solution to a given problem.” David Budgen in Software Design: An Introduction.

* 1. **Design Process**

The Design Process Software design is an iterative process that is traceable to the software requirements analysis process. Many software projects iterate through the analysis and design phases several times. Pure separation of analysis and design may not always be possible.

* 1. **Design Concepts**
* The design should be based on requirements specification.
* The design should be documented (so that it supports implementation, verification, and maintenance).
* The design should use abstraction (to reduce complexity and to hide unnecessary detail).
* The design should be modular (to support abstraction, verification, maintenance, and division of labor).
* The design should be assessed for quality as it is being created, not after the fact.
* Design should produce modules that exhibit independent functional characteristics.
* Design should support verification and maintenance.
  1. **Software Architecture**

Software architecture is a description of the subsystems and components of a software system and the relationships between them. You need to develop an architectural model to enable everyone to better understand the system, to allow people to work on individual pieces of the system in isolation, to prepare for extension of the system and to facilitate reuse and reusability.

* 1. **Describing an Architecture Using UML**

All UML diagrams can be useful to describe aspects of the architectural model. Four UML diagrams are particularly suitable for architecture modeling:

* Package diagrams
* Subsystem diagrams
* Component diagrams
* Deployment diagrams
  1. **Specifying Classes**

Each class is given a name, and then you need to specify:

* Attributes: initially those that capture interesting object states. Attributes can be public, protected, private or friendly/package.
* Operations: can be delayed till later analysis stages or even till design. Operations also can be public, protected, private or friendly/package.
* Object-Relationships:
* Associations: denote relationships between classes.
* An aggregation: a special case of association denoting a “consists of” hierarchy.
* Composition: a strong form of aggregation where components cannot exist without the aggregate.
* Generalization relationships: denote inheritance between classes.

**Lab Task:**

1. Draw Class Diagram for ATM machine
2. Draw Class Diagram for Library Management