Concept 1 : Recursion in Practice	I used this concept in the operation of LinkedList structure. In the PlayNode class, which is a node in a linked list data structure. For example, from line 26 - line 30, the countHelp method is recursive as it calls iteself(i.e., nextNode.countHelp(acc+1) for the next node until it reaches the end of the list. In PlayList class, the linked list use the method in the node class to do the recursion.
Concept 2 : Abstract Classes and Interfaces	I used this concept to build an AbstractTreeNode abstract class, a BasketBallAction class, a IList interface(this is an interface for the linkedlist) a INodei interface(this is an interface for the nodes in the linkedlit), a PlayStats interface(this is the interface for the tree hierarachical structure), a TreeNode interface for the node in the tree hierarchical structure), a PlayMode interface, and etc..
Concept 3 : Abstracted Linked Lists	I used this linkedlist data structure data to store the player's data in the game dynamically. I create a PlayList class for the linkedlist; a PlayNode class and a Empty Node class for the node in the linkedlist. And I also create a LinkedListLmp class as a driver class for the usage of this linkedlist structure.
Concep 4 : Higher order functions map, filter, and fold	I used this concept in the implementation of the linkedlist structure. I implemented these three advanced methods in the PlayerNode class from line 62 - line 81, and in the PlayList class from line 48 - line63. In the driver file called LinkedListImp class from line 65 - line 100.
Concept 5 : Hierarchical Data Representation	I used this concept to create a tree hierarchical data structure. I create a PlayStatsImp class which is the tree itself; GroupNode class and LeafNode class which both represent the node in this tree. And I create a driver file called PlayStatsDriver to help me store and manage each player's statistics in each branch.
Concept 6 : MVC Design	I used this concept to help me run a basketball game successfully. I create a Model class called BasketBallModel1, a Controller class called BasketBallControllerImp, a View class called BasketBallView1.
Concept 7 : SOLID Design Principles	I used this concept in the create of my interface and abstract classes. Single Responsibility Principle (SRP): Each class seems to have a single responsibility. For instance, StealAction, ShootAction, and BlockAction each handle a specific type of basketball action. Open/Closed Principle (OCP): The classes appear to be open for extension but closed for modification. For example, BasketBallAction is an abstract class extended by StealAction, ShootAction, and BlockAction. This design allows new actions to be added without modifying the existing code. Liskov Substitution Principle (LSP): The derived classes like StealAction, ShootAction, and

	BlockAction seem to be substitutable for their base class BasketBallAction. This suggests that they are following LSP, as instances of the base class can be replaced with instances of the derived classes without affecting the program's correctness. Interface Segregation Principle (ISP): The PlayMode interface is designed to be specific and client-driven, ensuring that classes like DefenseMode and OffenseMode that implement it are not forced to depend on methods they do not use. Dependency Inversion Principle (DIP): The use of interfaces (like PlayMode) and abstract classes (like BasketBallAction) suggests a design that depends on abstractions rather than concrete implementations. This indicates adherence to DIP, as high-level modules are not directly dependent on low-level modules.
Concept 8 : Any design pattern	For this concept, I used the Iterator Pattern to help me go through every action happened during the mini basketball game. I create a BasketBallActionIterator class which is the iterator itself. In the BasketBallGameControllerImp class from line 39 - line48 stands for the implementation of how we will use this iterator pattern.
Extension Concepts :	1. I create three driver classes: LinkedListImp class for the LinkedList, PlayerStatsDriver for the tree Hierarchical Structure, and the BasketBallGameDriver class for the MVC pattern. 2. I create some test classes: BasketBallGameModel1 class, PlayerStatsTest class, PlayerTest class, Statistics Test