Lab Exercise-3

Advanced Programming Lab-3

1. **Make your first inheritance based design**

   There were Pandavs and Kauravs. Arjun and Bheem were Pandavs. Duryodhan was a Kaurav. Pandavs were characterized by their skills of fighting (fight ( )), obedience (obey ( )), and kindness (**kind** ( )). [Note: kind is an adjective not a verb, so it should not be used a function, but let us bear with it for this problem] Though Bheem was little less kind but equally obedient as Arjun. Kauravs were notorious for disobeying and cruelty. But Kauravs were fighters. Kauravs were 100 in numbers, but one of them 'Vikarn' was a noble man- a good fighter, kind and obedient. If you dive a little more in the history, you will come to know that Pandavs and Kauravs were actually Bharatvanshi. And all bharatvanshis had been fighters.

   You are required to create a design first on paper and then write implementation on your machine. Make abstract classes and concrete classes. Decide which methods should be made abstract or non-abstract.

2. **Interfaces to be implemented by first non-abstract class**

   Make an interface 'Testable' that contains a method declaration for display ( ).

   a) Create a class 'Test' that implements Testable. Compile this class, and write your observations.
   b) Create another abstract class 'AbsTest' implements Testable. Compile this class, and write your observations.

   Note: Did you notice that methods in an interface are 'public' by default?

3. **Beautiful Code: Separate out uncompromisers**

   JUET students create a game of ducks. There were Rubber Ducks (RD), Wooden Ducks (WD), RedHead Ducks (RHD), and Lake Ducks (LD), as of now. There may be more in the future. As it can be guessed, all ducks can swim. RD and WD can't fly. RD squeaks. WD is mute. RHD and LD quack. Design and Implement.

   You might be tempted to make an abstract class 'Duck' by keeping fly and quack behaviors abstract and swim behavior defined. Although this approach is not wrong, but definitely not the best. Think and write some beautiful code using interfaces.
   Hint: Separate out uncompromisers.