3 Object-Oriented Programming (RKH)

(*a*) (*i*) Explain the purpose of access modifiers in OOP languages. [2 marks]

(*ii*) Copy and complete the table below to show the access restrictions for the four access modifiers in Java. [2 marks]

Access Modifier				
Defining class Class in same package Subclass in different package Non-subclass in different package				

(*b*) A Java game designer wishes to store all the game preferences (e.g., player name, screen size, music volume, etc.) within a custom `Preference` class.

(*i*) Assuming each preference is stored as a unique `String` key mapping to a `String` value, give a simple implementation of `Preference` that allows for efficiently setting or updating preferences and retrieving previously set ones. Your implementation should define an exception that is thrown when a preference key is requested but not present. [5 marks]

(*ii*) It is important that only one `Preference` object exists in a running game. Show how to apply access modifiers and the Singleton design pattern to ensure this. Your implementation should lazily instantiate the object. Is it necessary to make your class `final` or `Cloneable`? Explain your answer. [6 marks]

(*c*) The designer also implements other Singleton classes in the game and proposes to create a `SingletonBase` base class from which all such classes would inherit the singleton behaviour. By providing example Java code, explain why this is not viable. [5 marks]