To implement the trader and runes, I created various class. MerchantKale, Rune, SellAction, PurchaseAction, DeathAction, AttackAction and ConsumeAction.

Let’s start with MerchantKale. This Trader has an association with the Item class because his inventory list will consist of WeaponItems. The Merchant has dependencies with the PurchaseAction and SellAction because an instance of those classes will be created when the Player decides to trade with them. The merchant also has a dependency with the Player because they will be the one making the trades.

The PurchaseAction and SellAction has an association with the WeaponItem class, because a singular WeaponItem will be stored to sell or buy, this WeaponItem is passed into their constructor’s by MerchantKale.

Now for the Rune, the Rune is dropped during the DeathAction. The DeathAction is called from the AttackAction after checking if the target is conscious. The DeathAction has an association with the Rune, and one Rune will be dropped with a random value generated based on the enemy.

I think that this implementation is suitable for this task. The trading is handled solely by MerchantKale, and the Runes will be dropped during the DeathAction.