Based on the previously created Account class, implement the following use cases.

Now " **the account** *has a* **Policy.** "


```
Account

int : accountNumber
String : accountHolderName
int : balance
String : location

Policy : policy
```

```
Policy

int : policyNumber
String : policyName
int : premiumAmount
int : sumAssuredAmount
```
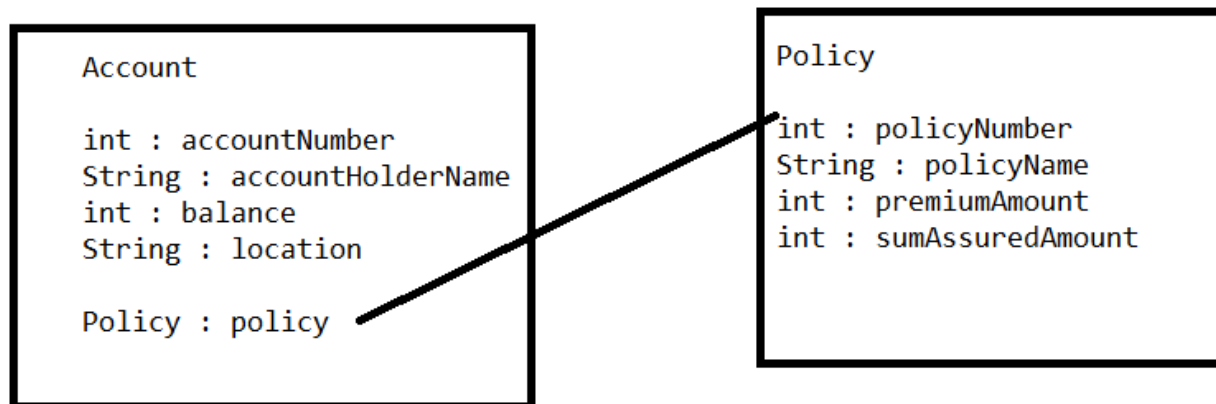
Maintain a relationship where one account has only one policy

a) Write code to insert account object in List<Account>
b) Write code to display the account based on the account number, if the account has no policy then in place of Policy details it should be "No Policy Attached".
c) Write code to get Account details based on policyName.
d) Write code to get Account details based on policyNumber.
e) Write code to get All accounts that have a policy *not attached.*
f) Write code to get the accounts where policy sumAssured is above a certain amount.(i.e. 100,000/- entered by the user)
g) Write code to get the accounts where the policy premium is below a certain amount. .(i.e. 10,000/- entered by user)
h) Get all the user who has policy belonging to a certain location (such as Banglore, entered by the user).
i) Get all accounts based on PolicyName , so store accounts in a Map<String , List<Account>> , where String is policyName