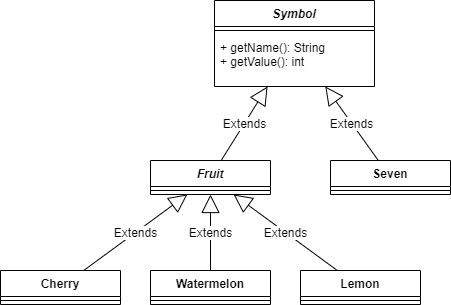
1. 撰寫Symbol.java及其子類別，繼承關係如下圖:



題目需求:

* 1. Symbol和Fruit為抽象類別(Abstract Class)，且該類別所有方法皆為抽象方法(Abstract Method)。
  2. 依照上圖之繼承關係實作所有類別。
  3. 實作子類別的getName方法，該方法需回傳該類別名(型態為String)。
  4. 實作子類別的getValue方法，該方法需回傳該類別的獎勵金額(型態為int)

|  |  |  |
| --- | --- | --- |
| **類別名稱** | **name** | **value** |
| Cherry | Cherry | 20 |
| Watermelon | Watermelon | 30 |
| Lemon | Lemon | 40 |
| Seven | Seven | 100 |

1. 撰寫一個Player.java   
   資料成員(instance variable):

|  |  |
| --- | --- |
| 資料名稱 | 資料型態 |
| name(姓名) | String |
| balance(餘額) | int |

play方法

|  |
| --- |
| public void play () {  …  } |

題目需求:

* 1. 該類別須包含上述資料成員。
  2. 基於物件導向設計封裝原則，上述資料成員將不對外公開，即外部物件無法直接存取上述資料。
  3. 為上述的每個資料成員，撰寫相對應的 get和set方法。
  4. 撰寫一個對姓名(name)及餘額 (balance)做初始化的建構子(constructor)。
  5. 實作play方法
     + play方法被呼叫時扣除5塊餘額(balance)。
     + 需隨機產生3個Symbol的物件，並將這三個物件的name印出。
     + 若生成的三個Symbol物件的name相同，則獲得獎勵，即餘額加上該類別的Value，並且印出獲得的獎勵和餘額。
     + 印出訊息格式如下:

|  |
| --- |
| Cherry, Cherry, Cherry  Reward: 20  Balance: 50 |

1. 撰寫一個 Main.java 類別，該類別只有main method，請在main method中實作以下功能:
   1. 使用者由console輸入兩組字串，分別為名稱(name)及餘額(balance)。
   2. 用上述建構子生成出一個玩家物件。
   3. 呼叫play方法。
2. 撰寫Account.java如下:

|  |
| --- |
| public class Account {  private String personId;  private float balance;  public String getPersonId() {  return personId;  }  public void setPersonId(String personId) {  this.personId = personId;  }  public float getBalance() {  return balance;  }  public void setBalance(float balance) {  this.balance = balance;  }  } |

1. 撰寫Bank.java如下:

|  |
| --- |
| public class Bank {  private List<Account> accounts = new List<Account>();  public boolean createAccount(String personId) {  // 1. 建立一個新帳號(Account)物件  // 2. 新的Account物件其balance的值為100  // 3. 檢查list長度  // 4. 當list長度大於等於10時，則回傳false，否則回傳true  // 5. 若list長度小於10，則將新的account加入list中  }  public float withdraw(String personId, float money) {  // 1. 從accounts中找到同personId的Account物件  // 2. 從該物件的balance減到提款的金額，即money  // 3. 回傳更新後的balance值  // 4. 若更新後的balance值小於0，則回傳-1  }  public float deposit(String personId, float money) {  // 1. 從accounts中找到同personId的Account物件  // 2. 更新該物件的balance的值，即加上存款的金額(money)  // 3. 回傳更新後的balance值  }  public List<Account> getAccounts() {  // 1. 回傳accounts  // 2. 注意外部只能讀此Bank物件中所有的Account，但無法修改  }  } |

題目需求:

* 1. 實作每一個方法中的註解需求。