2. In the code below, you will see an `Employee` class, as well as classes for three different types of bank accounts (`RetirementAccount, SavingsAccount, CheckingAccount`). An `Employee` has instance variables `id` and `accounts` (which is an array of accounts). Each employee account will always be one of the three account types mentioned above. There is also an `AccountManager` class containing an *unimplemented* static method `computeAccountBalanceSum`, which takes as input an array of `Employee` objects; for each such `Employee` object, `computeAccountBalanceSum` should extract the balance from each of the accounts in the array of accounts contained in that `Employee`, and add them to a running `sum` variable; finally, `computeAccountBalanceSum` should return the final value of `sum`.

   The objective of this problem is to implement `computeAccountBalanceSum` so that computation makes use of polymorphism. To do this, you will need to add a new class or interface that will generalize the different account objects; you will need to adjust the type of the array of accounts (currently it is of type `Object[]`) stored in `Employee` objects accordingly.

   NOTE: A method `getBalance` has already been implemented in each of the account classes – this method should be accessed in your polymorphic computation that is done in `computeAccountBalanceSum`.

   What you need to do:
   1. All `Object` arrays that can be found in the Employee class must be converted to the correct type. (You can just make corrections in the code provided.)
   2. Your implementation of `computeAccountBalanceSum` in `AccountManager` must correctly output the sum of the balances of all accounts in all the `Employee` objects passed in as an argument.
   3. Your implementation of `computeAccountBalanceSum` must make correct use of polymorphism .
   4. You are allowed to modify declarations of the different bank account classes, but the *final* keyword used in these classes may not be removed. (Just make changes directly in the code provided.)
   5. Wherever inheritance is needed for your polymorphic computation, it should be shown clearly in the code (this will require you to make some small corrections to the code provided).

```java
public class Employee {
    public Employee(String id, Object[] accounts) {
        this.id = id;
        this.accounts = accounts;
    }
    String id;
    Object[] accounts;
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public Object[] getAccounts() {
        return accounts;
    }
    public void setAccounts(Object[] accounts) {
        this.accounts = accounts;
    }
}
```

```java
public final class RetirementAccount {
    public RetirementAccount(double balance,
            LocalDate creationDate) {
        this.balance = balance;
        this.creationDate = creationDate;
    }
    private double balance;
    private LocalDate creationDate;
    public double getBalance() {
        double newbalance =
            balance - earlyWithdrawalFee();
        return newbalance < 0 ? 0: newbalance;
    }
    private double earlyWithdrawalFee() {
        return 50;
    }
}
```

```java
public final class CheckingAccount {
    public CheckingAccount(double balance) {
        this.balance = balance;
    }
    private double balance;
    public double getBalance() {
        return balance;
    }
}
```

```java
public final class SavingsAccount {
    public SavingsAccount(double balance,
            double interestRate) {
        this.balance = balance;
        this.interestRate = interestRate;
    }
    private double balance;
    private double interestRate;
    public double getBalance() {
        double newbalance
           = balance + interestRate * balance;
        return newbalance;
    }
}
```

```java
public class AccountManager {
    public static double computeAccountBalanceSum(Employee[] emps) {
    //SOLUTION:
        double sum = 0.0;
        for(Employee e : emps) {
           Account[] accts = getAccounts();
           for(Account a: accts) {
              sum += a.getBalance();
           }
        }
        return sum;
    }
```

```java
abstract class Account {
   abstract public getBalance();
}
```

GRADING:

7 points for correct implementation of computeAccountBalanceSum
3 points for correctly coding a superclass for the account classes
5 points for making all 5 changes to the original code (see below)

Modifications to original classes

```
public class Employee {
    public Employee(String id, Object[] accounts) {
        this.id = id;
        this.accounts = accounts;
    }
    String id;
    Object[] accounts; Account[] accounts;
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public Account[] getAccounts() {
        return accounts;
    }
    public void setAccounts(Object[] accounts) {
        this.accounts = accounts;
    }
}
```

```
public final class RetirementAccount extends Account {
    public RetirementAccount(double balance,
            LocalDate creationDate) {
        this.balance = balance;
        this.creationDate = creationDate;
    }
    private double balance;
    private LocalDate creationDate;
    public double getBalance() {
        double newbalance =
            balance - earlyWithdrawalFee();
        return newbalance < 0 ? 0: newbalance;
    }
    private double earlyWithdrawalFee() {
        return 50;
    }
}
```

```
public final class CheckingAccount extends Account {
    public CheckingAccount(double balance) {
        this.balance = balance;
    }
    private double balance;
    public double getBalance() {
        return balance;
    }
}
```

```
final class SavingsAccount extends Account {
    public SavingsAccount(double balance,
            double interestRate) {
        this.balance = balance;
        this.interestRate = interestRate;
    }
    private double balance;
    private double interestRate;
    public double getBalance() {
        double newbalance
            = balance + interestRate * balance;
        return newbalance;
    }
}
```