20210506_[程式閱讀]類別與物件

TestAccount.java

```
package com.object.account;

// 演示各種帳戶類的使用
public class TestAccount {

	public static void main(String[] args) {
		testBankAccount();
	}

	private static void testBankAccount() {
		BankAccount bank = new BankAccount(); // 創建一個銀行帳戶
		bank.saveCurrent(5000); // 活期帳戶存入 5000 元
		bank.saveDeposit(6, 5000); // 存入 6 個月定期的 5000 元
		System.out.println("第一次存款操作之後，"+bank.toString());
		bank.takeCurrent(2000); // 活期帳戶取出 2000 元
		bank.takeDeposit(6); // 取出 6 個月定期存款
		bank.saveDeposit(12, 5000); // 存入 12 個月定期的 5000 元
		System.out.println("第二次存款操作之後，"+bank.toString());
	}

}
```

_____.java

```
import java.util.Arrays;

//定義一個銀行帳戶類
public class BankAccount {
	private CashAccount current; // 活期存款帳戶
	private DepositAccount[] deposits; // 定期存款帳戶陣列

	// 銀行帳戶的構造方法
	public BankAccount() {
		current = new CashAccount(CashAccount.RMB, "活期存款");
		deposits = new DepositAccount[] {};
	}
```

```
// 往活期帳戶存入
public boolean saveCurrent(long amount) {
	return current.saveIn(amount);
}

// 從活期帳戶取出
public boolean takeCurrent(long amount) {
	return current.takeOut(amount);
}

// 往定期帳戶存入
public boolean saveDeposit(int depositTerm, long amount) {
	boolean result = false;
	int pos = getDepositPos(depositTerm); // 查找指定期限的定期帳戶
	if (pos >= 0) { // 已找到
		DepositAccount depositAccount = deposits[pos]; // 獲得已有的定期帳戶
		result = depositAccount.saveIn(amount); // 存入已有的定期帳戶
		deposits[pos] = depositAccount; // 更新已有的定期帳戶
	} else { // 未找到
		// 創建新的定期帳戶
		DepositAccount depositAccount = new DepositAccount(
				depositTerm, depositTerm + "個月定期存款");
		result = depositAccount.saveIn(amount); // 存入新的定期帳戶
		deposits = Arrays.copyOf(deposits, deposits.length + 1); // 陣列大小擴容
		deposits[deposits.length - 1] = depositAccount; // 插入新的定期帳戶
	}
	return result;
}

// 從定期帳戶取出
public boolean takeDeposit(int depositTerm) {
	boolean result = false;
	int pos = getDepositPos(depositTerm); // 查找指定期限的定期帳戶
	if (pos >= 0) { // 已找到
		DepositAccount depositAccount = deposits[pos]; // 獲得已有的定期帳戶
		// 取出已有的定期帳戶
		result = depositAccount.takeOut(depositAccount.getBalance());
		deposits[pos] = depositAccount; // 更新已有的定期帳戶
```

```
            }
            return result;
        }

        // 查找指定期限的定期帳戶
        private int getDepositPos(int depositTerm) {
            int pos = -1;
            for (int i = 0; i < deposits.length; i++) { // 遍歷定期帳戶陣列
                if (deposits[i].getDepositTerm() == depositTerm) { // 找到指定期限的定期帳戶
                    pos = i;
                    break;
                }
            }
            return pos;
        }

        // 輸出銀行帳戶的詳細資訊
        public String toString() {
            String desc = "銀行帳戶資訊如下：\n";
            desc = String.format("%s\t%s\n", desc, current.toString());
            for (DepositAccount item : deposits) { // 遍歷定期帳戶陣列
                desc = String.format("%s\t%s\n", desc, item.toString());
            }
            return desc;
        }
}
```

CashAccount.java

```
// 定義一個現金帳戶類
public class CashAccount extends Account {
    public final static int RMB = 0; // 人民幣
    public final static int SGD = 1; // 新加坡元
    public final static int USD = 2; // 美元
    public final static int EUR = 3; // 歐元
    public final static int GBP = 4; // 英鎊
    public final static int JPY = 5; // 日元
    public final static String[] typeNames = new String[]{"人民幣","新加坡元","美元","歐元","英鎊
","日元"};
    private int cashType; // 現金類型
```

```
    // 現金帳戶的構造方法
    public CashAccount(int cashType, String cashName) {
        super(cashName);
        this.cashType = cashType;
        setUnit(cashType==GBP ? "鎊" : "元"); // 設置餘額的單位
    }

    // 獲取現金的類型
    public int getCashType() {
        return this.cashType;
    }

    // 設置現金的類型
    public void setCashType(int cashType) {
        this.cashType = cashType;
    }

    // 輸出現金帳戶資訊
    public String toString() {
        String desc = String.format("現金類型為%s，%s", typeNames[this.cashType],
super.toString());
        return desc;
    }
}
```

DepositAccount.java

```
// 定一個定期存款帳戶類
public class DepositAccount extends Account {
    private int depositTerm; // 存款期限，單位元月

    // 存款帳戶的構造方法
    public DepositAccount(int depositTerm, String depositName) {
        super(depositName);
        this.depositTerm = depositTerm;
        setUnit("元"); // 設置餘額的單位
    }
```

```
    // 獲取存款的期限
    public int getDepositTerm() {
        return this.depositTerm;
    }

    // 設置存款的期限
    public void setDepositTerm(int depositTerm) {
        this.depositTerm = depositTerm;
    }

    // 這裡的定期存款只能一次性全部取出，不能分批取出，適合整存整取、零存整取。
    public boolean takeOut(long amount) {
        if (amount == getBalance()) { // 全部取出
            return super.takeOut(amount);
        } else {
            return false;
        }
    }

    // 輸出存款現金帳戶資訊
    public String toString() {
        String desc = String.format("存款期限為%d 個月，%s", this.depositTerm, super.toString());
        return desc;
    }
}
```

Account.java

```
//定義一個帳戶類
public class Account {
    private String name; // 帳戶名稱
    private long balance; // 帳戶餘額
    private String unit; // 餘額單位

    // 帳戶的構造方法
    public Account(String name) {
        this.name = name;
        this.balance = 0;
    }
```

```
// 獲取帳戶的名稱
public String getName() {
        return this.name;
}

// 設置帳戶的名稱
public void setName(String name) {
        this.name = name;
}

// 獲取帳戶的餘額
public long getBalance() {
        return this.balance;
}

// 設置帳戶的餘額
public void setBalance(long balance) {
        this.balance = balance;
}

// 獲取餘額的單位
public String getUnit() {
        return this.unit;
}

// 設置餘額的單位
public void setUnit(String unit) {
        this.unit = unit;
}

// 存入操作。返回 true 表示買入成功，false 表示買入失敗
public boolean saveIn(long amount) {
        if (amount > 0) {
                this.balance += amount; // 餘額增加
                return true;
        } else {
                return false;
```

```
        }
    }

    // 取出操作。返回 true 表示取出成功，false 表示取出失敗
    public boolean takeOut(long amount) {
        if (amount > 0) {
            this.balance -= amount; // 餘額減少
            return true;
        } else {
            return false;
        }
    }

    // 輸出帳戶資訊
    public String toString() {
        String desc = String.format("名稱為%s，餘額為%d%s", this.name, this.balance, this.unit);
        return desc;
    }

}
```