ตัวอย่างที่ ๑

```
๑. import java.io.Serializable;
๒. public class Account implements Serializable {
๓.   int no;
๔.   double balance;
๕.   public String toString() { return "Balance of account number " + no + "
   is " + balance; }
๖.   void deposit(double a) {
๗.     balance += a;
๘.   }
๙. }
```

บรรทัดที่ ๒ เป็นการกำหนดคลาส `Account` เพียงแค่บอกว่า `implements Serializable` ก็สามารถทำให้คลาส `Account` สามารถทำ serialize ได้ โดยไม่ต้องเขียนเมธอดใดเพิ่ม

ตัวอย่างที่ ๒

```
๑. import java.io.*;
๒. class Serialization๑ {
๓.   public static void main(String args[])throws IOException {
๔.     Account acc =new Account();
๕.     acc.deposit(5000);
๖.     System.out.println(acc);
๗.  FileOutputStream  fos =new FileOutputStream("Serialization๑.ser");
๘.  ObjectOutputStream oos=new ObjectOutputStream(fos);
๙.  oos.writeObject(acc);
๑๐.    }
๑๑. }
```

บรรทัดที่ ๓ ซึ่งกำหนดเมธอด main ต้องระบุ `throws IOException` เนื่องจากในคลาสนี้มีการใช้แฟ้มข้อมูล ซึ่งหากเกิดข้อผิดพลาดขึ้นจะมีการโยนสิ่งผิดปรกติ (exception) ออกไปนอกเมธอด `main`

บรรทัดที่ ๗ สร้างอ๊อบเจ็กท์ของคลาส `FileOutputStream` โดยส่งชื่อแฟ้มที่ต้องการให้เปิดเข้าไปเป็นพารามิเตอร์

บรรทัดที่ ๘ สร้างอ๊อบเจ็กท์ของคลาส `ObjectOutputStream` ซึ่งเป็นคลาสที่จะใช้สำหรับทำ serialize โดยส่งค่าอ้างอิงที่อ้างถึงอ๊อบเจ็กท์ของ `FileOutputStream` ไปให้เป็นพารามิเตอร์เพื่อบอกให้รู้ว่าเราต้องการให้ส่งข้อมูลไปเก็บในแฟ้มใด

บรรทัดที่ ๙ เป็นการเขียนอ๊อบเจ็กท์ `acc` ลงไฟล์

หากรันโปรแกรมนี้ จะได้แฟ้มข้อมูลชื่อ serialization1.ser ซึ่งจะมีการเก็บเลข version และตัวแปรสมาชิกของคลาส `Account` ขนาดของแฟ้มที่ได้เป็น ๕๕ ไบต์

ตัวอย่างที่ ๓

```
๑. import java.io.*;
๒. class Deserialization1 {
๓.   public static void main(String args[])throws IOException,
   ClassNotFoundException {
๔.   FileInputStream  fis =new FileInputStream("Serialization1.ser");
๕.   ObjectInputStream ois=new ObjectInputStream(fis);
๖.   Account acc =(Account)ois.readObject();
๗.   System.out.println(acc);
๘.   }
๙. }
```

ตัวอย่างที่ ๓ เป็นตัวอย่างโปรแกรมที่ทำ deserialize จะนำข้อมูลของอ๊อบเจ็กท์ที่ save ไว้ที่แฟ้ม serialization1.ser กลับมาสร้างเป็นอ๊อบเจ็กท์อีกครั้ง

โปรแกรมนี้มีการใช้คลาส Account โดยที่ไม่ได้ใส่คลาสนี้ไว้ใน source code ด้วย แต่จะอาศัยคลาส `Account` ที่ได้จากการคอมไพล์ตัวอย่างที่ ๑

บรรทัดที่ ๓ เป็นเมธอด `main` มีการใส่ สิ่งผิดปรกติเพิ่มอีก 1 ตัวคือ `ClassNotFoundException` เนื่องจากบรรทัดที่ ๖ มีการใช้เมธอด `readObject` เมธอดนี้จะอ่านข้อมูลจากแฟ้มเข้ามาก ซึ่งในแฟ้มจะมีการเก็บข้อมูลไว้ว่าเป็นข้อมูลที่ save มาจากคลาสอะไร เพื่อให้เมื่ออ่านข้อมูลเข้ามามันจะรู้ได้ว่าจะต้องสร้างอ๊อบเจ็กท์ของคลาสอะไรขึ้นมาก่อนที่จะนำข้อมูลจากแฟ้มกับคืนไปใส่ไว้ในตัวแปรสมาชิก

บรรทัดที่ ๔ เป็นการเปิดแฟ้มที่ต้องการอ่านข้อมูลที่ save ไว้

บรรทัดที่ ๕ เป็นการสร้างอ๊อบเจ็กท์จากคลาส `ObjectInputStream` โดยส่งค่าอ้างอิงถึงอ๊อบเจ็กท์ของคลาส `FileInputStream` ที่ได้จากบรรทัดที่ ๕ ไปให้เป็นพารามิเตอร์ อ๊อบเจ็กท์นี้จะอ้างถึงโดยตัวแปร `ois`

บรรทัดที่ ๖ มีการเรียกใช้เมธอด `readObject` ของอ๊อบเจ็กท์ `ois` (คลาส `ObjectInputStream`) เมธอดนี้จะรู้จากข้อมูลที่อ่านจากแฟ้มเข้ามาว่าเป็นข้อมูลที่ save มาจากคลาสอะไร มันจะสร้างอ๊อบเจ็กท์ของคลาสนั้นขึ้นมาใหม่ แล้วจึงอ่านข้อมูลจากแฟ้มต่อไปเพื่อนำข้อมูลจากแฟ้มไปใส่ในตัวแปรสมาชิกจนครบทุกตัวที่ save ไว้ในแฟ้ม เมื่อนำข้อมูลไปใส่ในตัวแปรครบทุกตัวที่ได้ save ไว้แล้ว มันจะคืนกลับโดยส่งที่อยู่ของอ๊อบเจ็กท์กลับออกไปด้วย แต่เนื่องจากเมธอด `readObject` ประกาศไว้ว่าคืนค่าเป็น คลาส `Object` จึงต้อง cast ให้เป็นคลาสที่เราต้องการคือ `Account`

ตัวอย่างที่ ๔

```
๑. import java.io.Serializable;
๒. public class Account2 implements Serializable {
๓.   int no;
๔.   String name;  // add a new member variable
๕.     double balance;
๖.     Account2(int no, String name) { this.no = no; this.name = name; }
๗.     public String toString() { return "Balance of account " + no +
```

```
๘.                 ""+name + " is " + balance; }
๙.     void deposit(double a) {
๑๐.            balance += a;
๑๑.    }
๑๒. }
```

ตัวอย่างที่ ๔ ปรับปรุงมาจากตัวอย่างที่ ๑ โดยเพิ่มตัวแปรสมาชิกให้กับคลาส Account อีก 1 ตัวคือตัวแปร name เป็น String ด้วยเจตนาที่จะแสดงให้เห็นถึงการ save อ๊อบเจ็กท์ที่มีการบรรจุอ๊อบเจ็กท์อื่นไว้ภายในว่ามันจะ save อ๊อบเจ็กท์อื่นที่เกี่ยวข้องให้ด้วย

เนื่องจากปรับปรุงคลาส Account ให้ต่างไปจากเดิมจึงเปลี่ยนชื่อคลาสเป็น Account2 เพื่อให้ผลลัพธ์จากการคอมไพล์ซึ่งจะได้แฟ้มคลาส Account ชื่อไม่ซ้ำกับคลาสที่ได้จากตัวอย่างที่ ๑ (ไม่ต้องการให้คลาส Account ใหม่ไปทับคลาส Account เดิม)

ตัวอย่างที่ ๕

```
๑. import java.io.*;
๒. class Serialization2 {
๓.   public static void main(String args[])throws IOException {
๔.     Account2 acc =new Account2(1234, "Test Account");
๕.     acc.deposit(5000);
๖.     System.out.println(acc);
๗.  ObjectOutputStream oos=new ObjectOutputStream(new
   FileOutputStream("Serialization2.ser"));
๘.  oos.writeObject(acc);
๙.   }
๑๐. }
```

ตัวอย่างที่ ๕ นี้ปรับปรุงมาจากตัวอย่างที่ ๒ โปรแกรมนี้เปลี่ยนชื่อแฟ้มข้อมูลที่ save อ๊อบเจ็กท์เป็น serialization2.ser เพื่อไม่ให้เขียนข้อมูลทับแฟ้มเดิม

เนื่องจากคลาส Account2 มีตัวแปร name เพิ่มขึ้นสำหรับเก็บชื่อบัญชี จึงได้เขียนตัวสร้างให้รับพารามิเตอร์เป็นเลขที่บัญชี และชื่อบัญชี เพื่อที่จะได้กำหนดค่าเริ่มต้นให้อ๊อบเจ็กท์ตั้งแต่แรกที่อ๊อบเจ็กท์ถูกสร้างขึ้นมา และปรับปรุงเมธอด toString ให้แสดงเลขที่และชื่อบัญชีด้วย

ในเมธอด main มีการสร้างอ๊อบเจ็กท์ของคลาส ObjectOutputStream เช่นเดิม แต่เขียนให้กระชับกว่าเดิม โดยรวมเอาการกำหนดแฟ้มที่จะเปิดไว้ในบรรทัดเดียวกันไปเลย โดยไม่ต้องเก็บพักที่อยู่ของอ๊อบเจ็กท์ของคลาส FileOutoutStream ไว้ในตัวแปรอ้างอิง

เมื่อรันโปรแกรมนี้ มันจะสร้างแฟ้มข้อมูลชื่อ serialization2.ser ขนาด ๙๙ ไบต์ จะเห็นว่าขนาดใหญ่กว่าแฟ้ม serialization1.ser ๔๔ ไบต์ อันเป็นผลมาจากการ save อ๊อบเจ็กท์ของคลาส String เพิ่มขึ้นนั่นเอง

ตัวอย่างที่ ๖

```
๑. import java.io.*;
๒. class Deserialization2 {
๓.   public static void main(String args[]) throws IOException,
   ClassNotFoundException {
๔.   FileInputStream  fis = new FileInputStream("Serialization2.ser");
๕.   ObjectInputStream ois= new ObjectInputStream(fis);
๖.   Account2 acc =(Account2)ois.readObject();
๗.   System.out.println(acc);
๘.    }
๙. }
```

ตัวอย่างที่ ๖ เป็นโปรแกรมที่อ่านข้อมูลจากแฟ้ม Serialization2.ser กลับมาสร้างเป็นอ๊อบเจ็กท์ของคลาส Account2 อีกครั้ง โปรแกรมนี้คล้ายกับตัวอย่างที่ ๓ ต่างกันที่ชื่อแฟ้มที่จะอ่าน และชื่อคลาสที่ทำ deserialize

ตัวอย่างที่ ๗

```
๑๐. import java.io.*;
๑๑. class Serialization3 {
๑๒.   public static void main(String args[]) throws IOException {
๑๓.     Address addr =new Address("333/444", "Prachan Road", "Bangkok", 10200);
๑๔.     Account3 acc =new Account3(1234, "Test Account", addr);
๑๕.     acc.deposit(5000);
๑๖.     System.out.println(acc);
๑๗.     ObjectOutputStream oos= new ObjectOutputStream(new
   FileOutputStream("Serialization3.ser"));
๑๘.     oos.writeObject(acc);
๑๙.   }
๒๐. }
๒๑.
๒๒. class Account3 implements Serializable {
๒๓.   int no;
๒๔.   String name;
๒๕.   Address contact; //add a new member variable
๒๖.   double balance;
๒๗.   Account3(int no, String name, Address a){ this.no =no; this.name =name;
   contact =a; }
๒๘.   public String toString(){ return "Balance of account "+no +""+name +"is "
   +balance; }
๒๙.   void deposit(double a){
๓๐.     balance +=a;
๓๑.   }
๓๒. }
๓๓.
๓๔. class Address {
๓๕.   String homeAddress;
```

```
๓๖.    String streetAddress;
๓๗.    String province;
๓๘.    short  postalCode;
๓๙.    Address(String h, String s, String p, int pcode){
๔๐.      homeAddress =h; streetAddress =s; province=p; postalCode =
   (short)pcode;
๔๑.    }
๔๒. }
```

ตัวอย่างที่ ๗ เพิ่มคลาส `Address` ซึ่งจะให้คลาส `Account3` บรรจุคลาสนี้ไว้ด้วย แต่คลาส `Address` ไม่ได้ `implement Serializable` เมื่อรันโปรแกรมนี้จะเกิดสิ่งผิดปรกติ `NotSerializableException` ถ้าจะให้รันโปรแกรมนี้ได้ จะต้องแก้ไขอย่างไร?