Lập trình mạng - Lab 1.1. Java Serialization

Tuần tự hoá (serialization) đối tượng là tiến trình lưu trạng thái của đối tượng thành một chuỗi các byte; cung như tiến trình xây dựng lại những byte này thành đối tượng. Java Serialization API cung cấp các phương thức chuẩn cho các nhà phát triển thực hiện tuần tự hoá đối tượng.

Resources

- Discover the secrets of the Java Serialization API article from java.sun.com
- Object Serialization tutorial from java.sun.com
- Java Serialization tutorial from mactech.com

Exercises

- Exercise 1: Serialize and deserialize an object
- Exercise 2: Customizing default protocol
- Exercise 3: Buffered stream
- Luyện tập

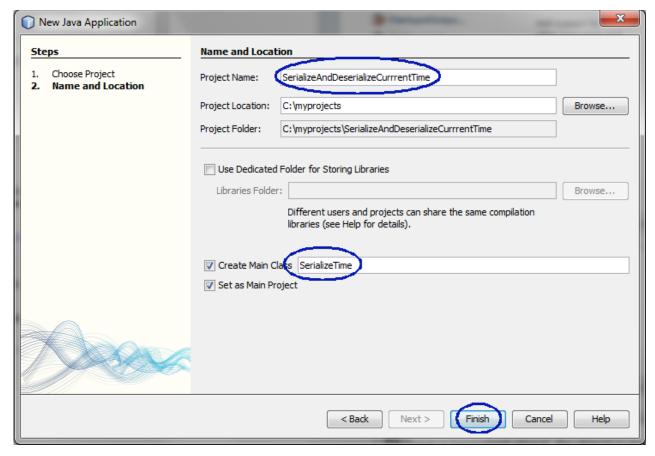
Exercise 1: Serialize and Deserialize an object

Trong bài lab này, chúng ta sẽ làm quen với tuần tự hoá (serialization) và thôi tuần tự hoá (deserialization) một đối tượng, cũng như cách dùng từ khoá transient.

- 1. Serialize the current time
- 2. Use transient keyword

(1.1) Tuần tự hoá thời điểm hiện tại

- 0. Khởi động NetBeans IDE.
- 1. Tạo mới một project NetBeans
 - Chon File->New Project (Ctrl+Shift+N).
 - Cửa sổ **New Project** xuất hiện
 - Tai pane Choose Project, chon Java trong Categories và Java Application trong Projects.
 - Click Next.
 - Tại pane Name and Location, với trường Project Name, gỗ tên project SerializeAndDeserializeCurrrentTime
 - Với trường Create Main Class, gõ SerializeTime.
 - Click Finish.



- Project SerializeAndDeserializeCurrrentTime và file SerializeTime.java xuất hiện trong editor nguồn của IDE NetBeans.
- 2. Sửa file **SerializeTime.java** như đoạn Code-1.11. Chú ý nhưng dòng code in đỏ.

```
import java.io.ObjectOutputStream;
import java.io.FileOutputStream;
import java.io.IOException;
public class SerializeTime{
  public static void main(String [] args){
     String filename = "time.ser";
     if(args.length > 0) {
       filename = args[0];
     // Create an object
     PersistentTime time = new PersistentTime();
     // Serialize the object instance and save it in
     // a file.
     FileOutputStream fos = null;
     ObjectOutputStream out = null;
     try {
       fos = new FileOutputStream(filename);
       out = new ObjectOutputStream(fos);
       out.writeObject(time);
       out.close();
     } catch(IOException ex) {
       ex.printStackTrace();
     System.out.println("Current time is saved into " + filename);
  }
```

```
SerializeAndDeserializeCurrrentTime - NetBeans IDE 6.9
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
                                                              !!! • (1) •
                                                                           Q - | Search (Ctrl+I)
                             <default config>
                               ...age 📝 SQL Command 45 🔞 📝 SQL Command 46 🕸 🚳 SerializeTime.java 🔞
      Services
SerializeAndDeserializeCurrrentTil
                                    B - 3 - | C 등 주 등 등 | 살 살 | 🔘 🗆 | 🕮 🚅
A .
                                 9
   i- (default package >
                                10
                                      public class SerializeTime {
      SerializeTime.java
                                11
 🗓 - 🚹 🏻 Test Packages
                                12 🖃
 🕁 ┣ Libraries
                                13
                                            * @param args the command line arguments
🖮 🍃 Test Libraries
                                           */
                                14
                                15 🖃
                                          public static void main(String[] args) {
                                                       String filename = "time.ser";
                                16
                                 17
                                               if(args.length > 0) {
                                                   filename = args[0];
                                 18
                                19
                                 20
                                               // Create an object
                                 21
                                 8
                                               PersistentTime time = new PersistentTime();
                                 23
                                               \ensuremath{//} Serialize the object instance and save it in
                                 24
                                 25
                                               // a file.
                                26
                                               FileOutputStream fos = null;
                                               ObjectOutputStream out = null;
                                 27
                                 28
                                               try {
                                                   fos = new FileOutputStream(filename);
                                 29
                                 30
                                                   out = new ObjectOutputStream(fos);
                                                   out.writeObject(time);
                                 31
                                                   out.close();
                                 32
                                 33
                                               } catch(IOException ex) {
                                 <u>@</u>
                                                   ex.printStackTrace();
                                 35
                                 36
                                               System.out.println("Current time is saved into "
                                 37
                                 38
                                 39
                                                                                                      Ŧ
                                 40
                                                                                                 5 | 1 INS
```

3. Viết lớp **PersistentTime.java**.

```
import java.io.Serializable;
import java.util.Date;
import java.util.Calendar;

public class PersistentTime implements Serializable{
    private Date time;

    public PersistentTime() {
        time = Calendar.getInstance().getTime();
    }

    public Date getTime() {
        return time;
    }
}
```

```
SerializeAndDeserializeCurrrentTime - NetBeans IDE 6.9
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
                                                                       Q - | Search (Ctrl+I)
                           <default config>
                                                                                          4 → ▼ □
                             ... 46 🚳 SerializeTime.java 🛭 🚳 PersistentTime.java 🕱
     Services
SerializeAndDeserializeCurrrentTi
                              🚊 🚹 Source Packages
                               1 🖃 /*
  2
                                     * To change this template, choose Tools | Templates
      --- 🚳 PersistentTime.java
                               3
                                     * and open the template in the editor.
       SerializeTime.java
                               4
🗓 - 🚹 🏻 Test Packages
                               5
🗓 🍃 Libraries
                               6 import java.io.Serializable;
🖮 🔓 Test Libraries
                               7
                                  import java.util.Date;
                               8
                                 import java.util.Calendar;
                               9
                              10
                                   public class PersistentTime implements Serializable{
                              11
                                        private Date time;
                              12
                              13 🖃
                                        public PersistentTime() {
                                            time = Calendar.getInstance().getTime();
                              14
                              15
                              16
                              17 -
                                        public Date getTime() {
                                            return time;
                              18
                              19
                              20
                              21
                                                                                          21 | 1 INS
```

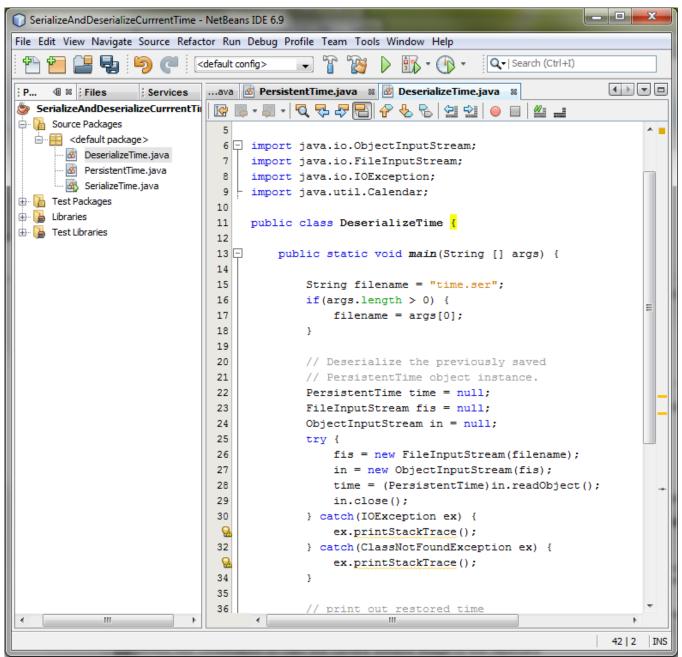
3. Viết lớp **DeserializeTime.java**.

```
import java.io.ObjectInputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Calendar;
public class DeserializeTime {
  public static void main(String [] args) {
     String filename = "time.ser";
     if(args.length > 0) {
        filename = args[0];
     // Deserialize the previously saved
     // PersistentTime object instance.
     PersistentTime time = null;
     FileInputStream fis = null;
     ObjectInputStream in = null;
     try {
       fis = new FileInputStream(filename);
```

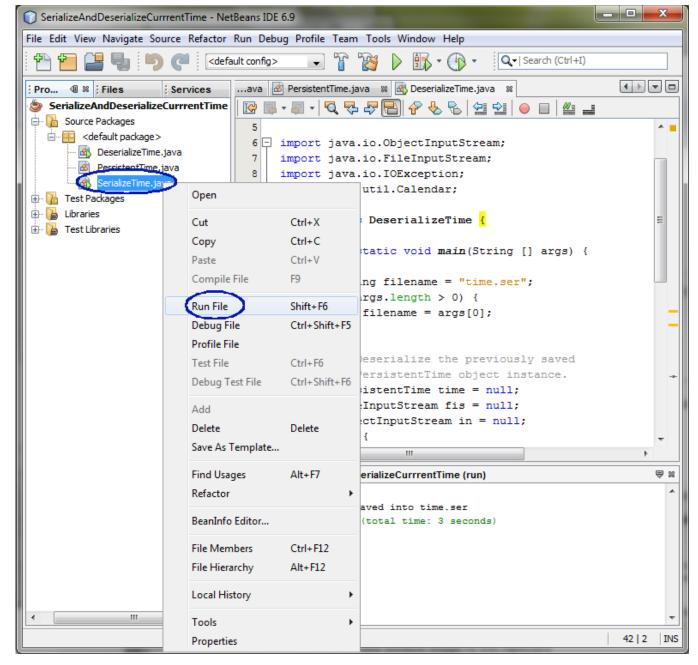
```
in = new ObjectInputStream(fis);
    time = (PersistentTime)in.readObject();
    in.close();
} catch(IOException ex) {
    ex.printStackTrace();
} catch(ClassNotFoundException ex) {
    ex.printStackTrace();
}

// print out restored time
System.out.println("Previously serialized time: " + time.getTime());

// print out the current time
System.out.println("Current time: " + Calendar.getInstance().getTime());
}
```



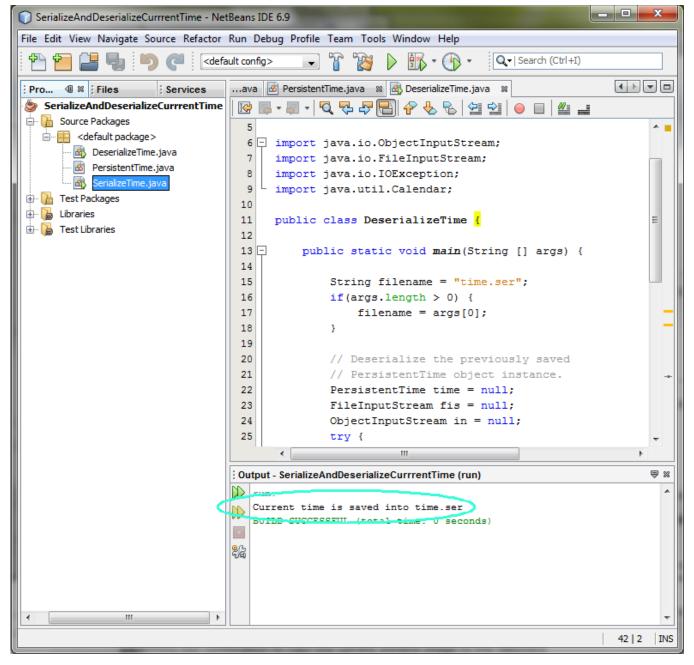
- 4. Biên dịch và chạy phần Serialization
 - Click chuột phải lên SerializeTime.java và chọn Run File.



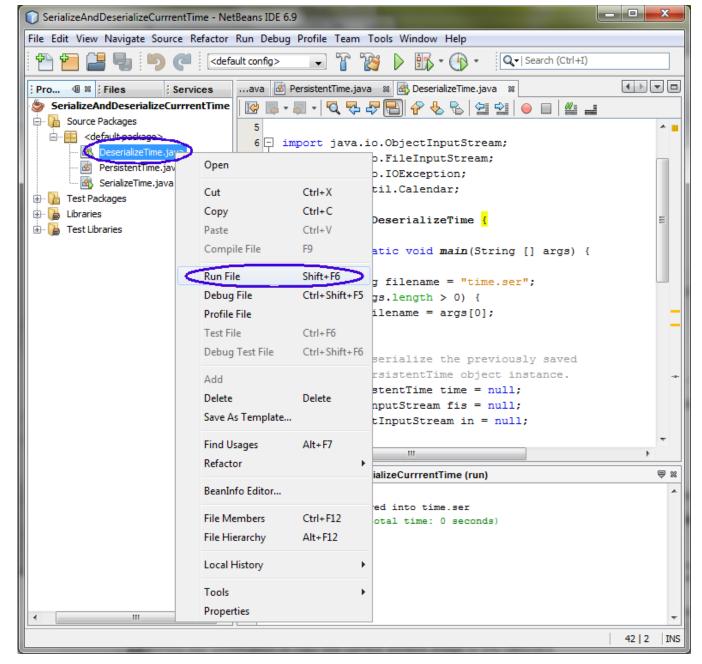
• Nghiên cứu kết quả trong cửa sổ **Output**. (Figure-1.13)

Current time is saved into time.ser

Figure-1.13: Result of running SerializeAndDeserializeCurrrentTime application



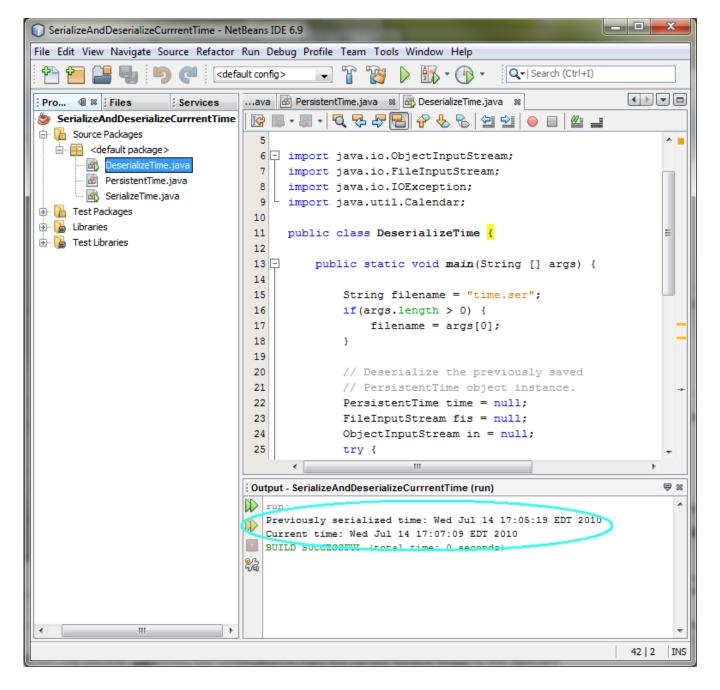
- 5. Biên dịch và chạy Deserialization.
 - Click chuột phải lên file DeserializeTime.java và chọn Run File.



• Quan sát kết quả trong cửa sổ Output. (Figure-1.13)

```
Previously serialized time: Mon Feb 26 01:57:16 EST 2007
Current time: Mon Feb 26 01:58:58 EST 2007
```

Figure-1.13: Result of running SerializeAndDeserializeCurrrentTime application



5. Luyện tập.

- Thêm trường mới **myName** kiểu **String** cho lớp PersistentTime.
- Sửa file SerializeTime.java and DeserializeTime.java để hiện thị trường myName.

return to top of the exercise

(1.2) Sửa dụng từ khoá transient

1. Sửa PersistentTime.java như Code-1.11.

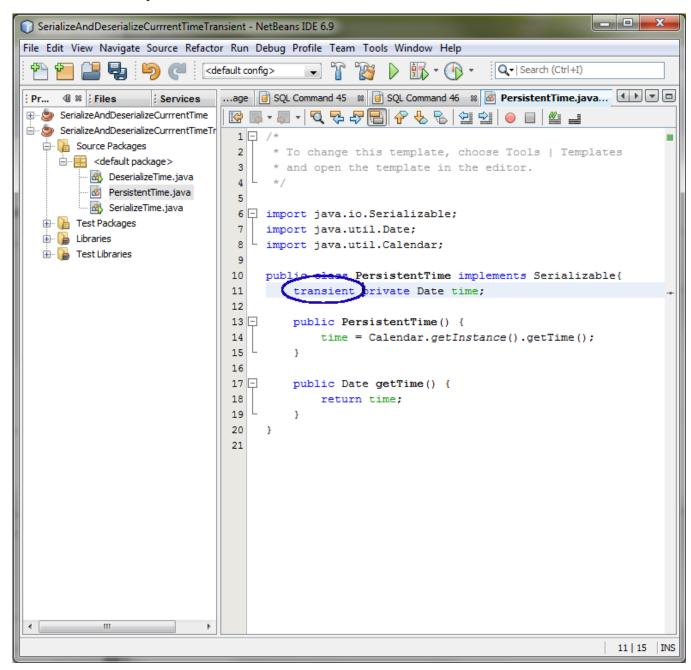
```
import java.io.Serializable;
import java.util.Date;
import java.util.Calendar;

public class PersistentTime implements Serializable{
    transient private Date time;

public PersistentTime() {
    time = Calendar.getInstance().getTime();
}
```

```
public Date getTime() {
    return time;
    }
}
```

Code-1.11: SerializeTime.java



- 2. Biên dịch và chạy Serialization
 - Click Chuột phải vào SerializeTime.java và chọn Run.
 - Xem kết quả xuất hiện trong cửa sổ **Output**. (Figure-1.13)

Current time is saved into time.ser

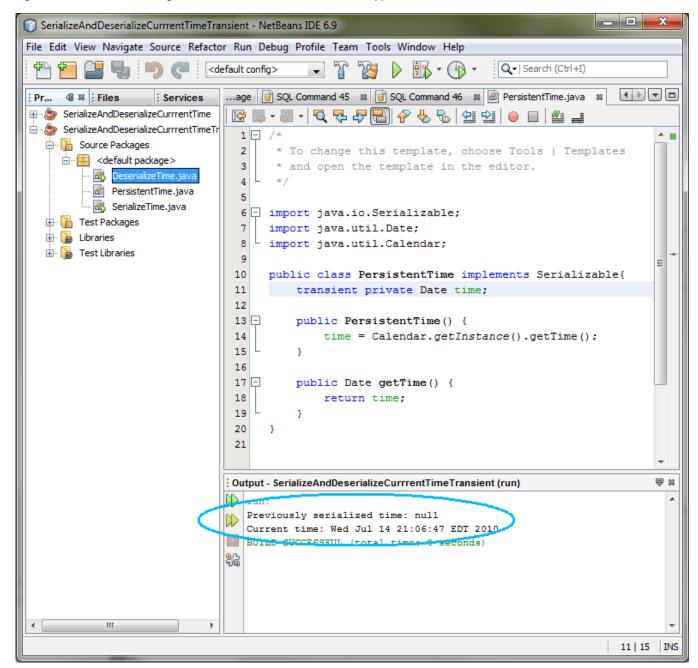
Figure-1.13: Result of running SerializeAndDeserializeCurrrentTime application

- 3. Biên dịch và chạy Serialization
 - Click Chuột phải vào **DeserializeTime.java** và chọn Run.
 - Xem kết quả xuất hiện trong cửa sổ **Output**. (Figure-1.13)

Previously serialized time: null

Current time: Mon Feb 26 02:07:09 EST 2007

Figure-1.13: Result of running SerializeAndDeserializeCurrrentTime application



4. Luyện tập thêm:

- Thêm trường myName với từ khoá transient.
- Test với trường vừa thêm.

Summary

Trong bài lab này, chúng ta đã làm quen với tuần tự hoá (serialization) và thôi tuần tự hoá (deserialization) một đối tượng, cũng như cách dùng từ khoá transient.

return to the top

Exercise 2: Version Control

Trong bài lab này, chúng ta sẽ học cách để kiểm tra phiên bản (version control).

- 1. Sửa lớp sau khi tuần tư hoá
- 2. Sử dụng id phiên bản duy nhất

(2.1) Sửa lớp sau khi tuần tự hoá

- 1. Biên dịch và chạy phần Serialization.
 - Click chuột phải lên SerializeTime.java và chọn Run File.
 - Quan sát kết quả trong cửa sổ **Output**. (Figure-2.10)

```
Current time is saved into time.ser
```

Figure-2.10: Result

2. Sửa file **PersistentTim.java** như trong Code-2.11. Thêm trường mới trong lớp **PersistentTime** sau khi tuần tự hoá.

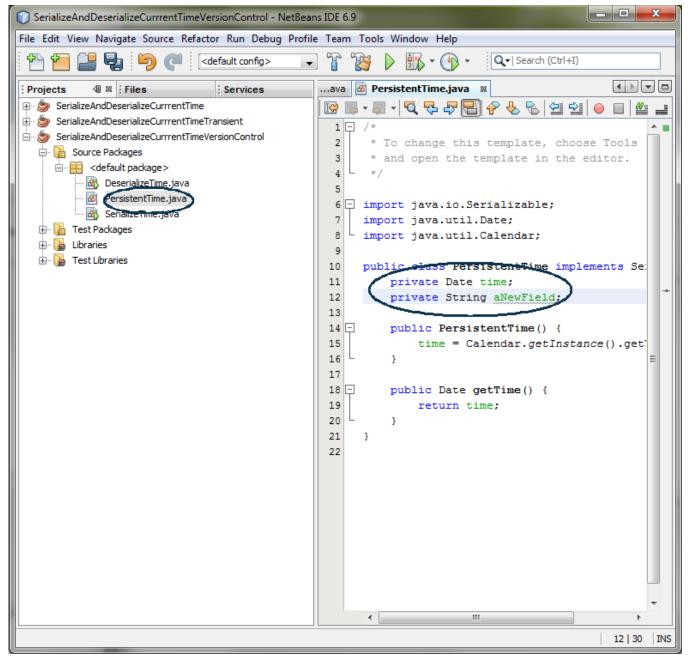
```
import java.io.Serializable;
import java.util.Date;
import java.util.Calendar;

public class PersistentTime implements Serializable{
    private Date time;
    private String aNewField;

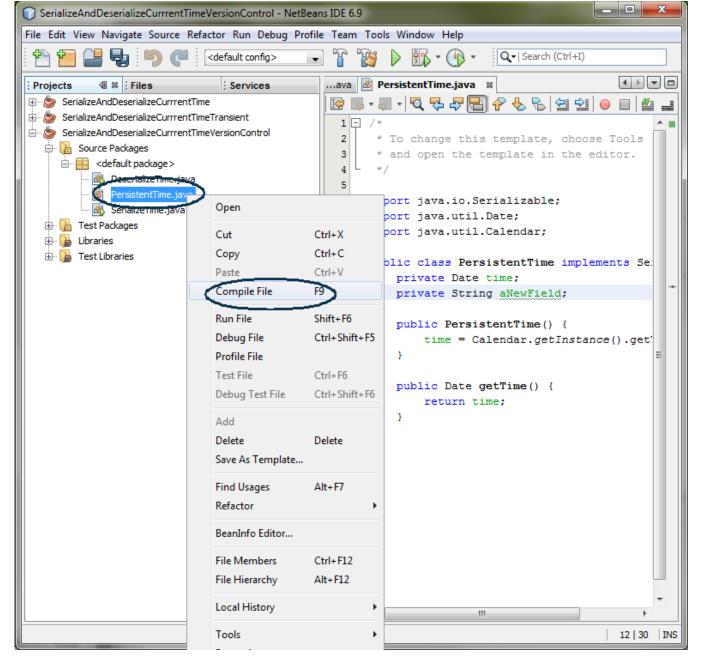
    public PersistentTime() {
        time = Calendar.getInstance().getTime();
    }

    public Date getTime() {
        return time;
    }
}
```

Code-2.11: FileReaderWriter.java



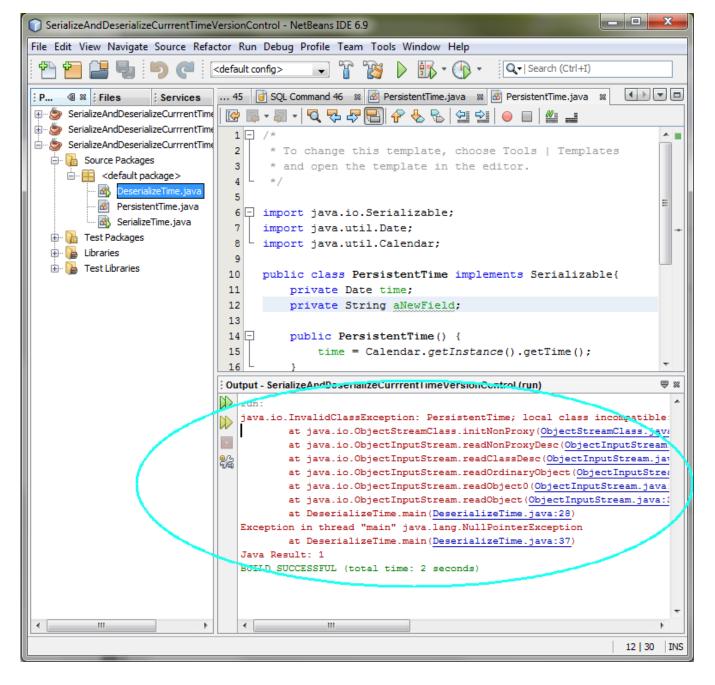
- 3. Biên dịch PersistentTime.java.
 - Click chuột phải lên PersistentTime.java và chọn Compile File.



- 4. Biên dịch và chạy Deserialization
 - Click Chuột phải vào **DeserializeTime.java** và chọn Run.
 - Ngoại lệ java.io.InvalidClassException xuất hiện trong cửa sổ Output. (Figure-2.12 below)

```
java.io.InvalidClassException: PersistentTime; local class incompatible: stream classdesc serialVersionUID =
-3126998878902358585, local class serialVersionUID = -5560460247034149373
    at java.io.ObjectStreamClass.initNonProxy(ObjectStreamClass.java:519)
    at java.io.ObjectInputStream.readNonProxyDesc(ObjectInputStream.java:1546)
    at java.io.ObjectInputStream.readClassDesc(ObjectInputStream.java:1460)
    at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1693)
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1299)
    at java.io.ObjectInputStream.readObject(ObjectInputStream.java:339)
    at DeserializeTime.main(DeserializeTime.java:23)
Exception in thread "main" java.lang.NullPointerException
```

Figure-2.12: java.io.InvalidClassException exception



(2.2) Use a version id

1. Sửa file **PersistentTim.java** như trong Code-2.11. Thêm id duy nhất cho lớp.

```
import java.io.Serializable;
import java.util.Date;
import java.util.Calendar;

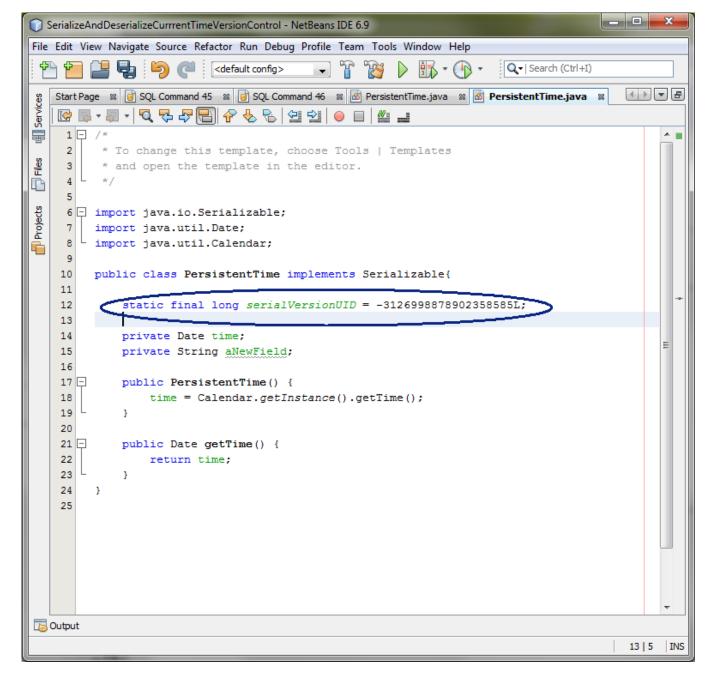
public class PersistentTime implements Serializable{

    static final long serialVersionUID = -3126998878902358585L;

    private Date time;
    private String aNewField;

    public PersistentTime() {
        time = Calendar.getInstance().getTime();
    }

    public Date getTime() {
        return time;
    }
}
```



2. Biên dịch PersistentTime.java.

- Click chuột phải PersistentTime.java và chọn Compile File.
- 3. Biên dịch và chạy Serialization
 - Click Chuột phải vào SerializeTime.java và chọn Run.
 - Xem xét kết quả trong cửa sổ **Output**. (Figure-1.13)

```
Current time is saved into time.ser
```

Figure-1.13: Result of running SerializeAndDeserializeCurrrentTime application

4. 2. Sửa file **PersistentTim.java** như trong Code-2.11. Thêm trường mới trong lớp **PersistentTime** sau khi tuần tự hoá.

```
import java.io.Serializable;
import java.util.Date;
import java.util.Calendar;
public class PersistentTime implements Serializable{
```

```
static final long serialVersionUID = -3126998878902358585L;

private Date time;
private String aNewField;

private String aNewNewField;

public PersistentTime() {
    time = Calendar.getInstance().getTime();
}

public Date getTime() {
    return time;
}
}
```

Code-2.11: FileReaderWriter.java

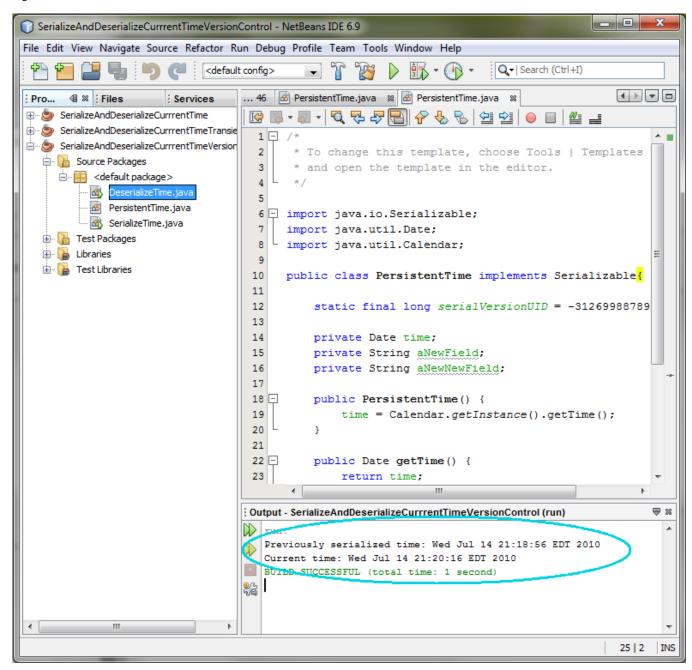
```
_ D X
SerializeAndDeserializeCurrrentTimeVersionControl - NetBeans IDE 6.9
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
                                                     Q+| Search (Ctrl+I)
            | <default config>
                                                                                       ←→ ▼ ₽
   Start Page 🔞 🗃 SQL Command 45 🔉 📑 SQL Command 46 💸 🚳 PersistentTime.java 💸 🚳 PersistentTime.java
   A .
          * To change this template, choose Tools | Templates
     2
     3
          * and open the template in the editor.
     4
5
Projects
     6 import java.io.Serializable;
        import java.util.Date;
       import java.util.Calendar;
     8
     9
    10
        public class PersistentTime implements Serializable{
    11
    12
             static final long serialVersionUID = -3126998878902358585L;
    13
    14
             private Date time;
    15
             private String aNewField;
    16
             private String aNewNewField;
    17
    18 🖃
             public PersistentTime() {
    19
                 time = Calendar.getInstance().getTime();
    20
    21
    22 -
             public Date getTime() {
    23
                 return time;
    24
    25
         }
    26
🔁 Output
                                                                                       25 | 2 INS
```

- 5. Biên dịch **PersistentTime.java**.
 - Click chuột phải PersistentTime.java và chọn Compile File.
- 6. Biên dịch và chạy Deserialization

- Click Chuột phải vào **DeserializeTime.java** và chọn Run.
- Xem xét kết quả xuất hiện trong cửa sổ **Output**.

Previously serialized time: Mon Feb 26 03:04:59 EST 2007 Current time: Mon Feb 26 03:05:13 EST 2007

Figure-2.23: Result



return to top of the exercise

Summary

Trong bài lab này, chúng ta sẽ học cách để kiểm tra phiên bản (version control) trong tuần tự hoá và thôi tuần tự hoá của một đối tương.

return to the top

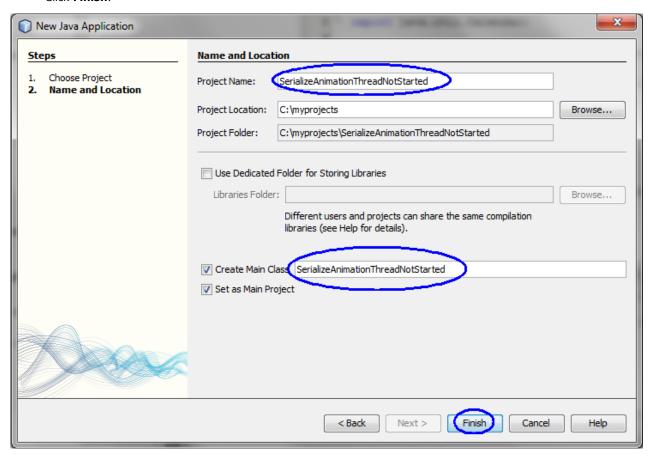
Exercise 3: Customizing default protocol

Trong bài lab này, chúng ta sẽ học cách tạo phương thức readObject() và writeObject() riêng.

- 1. Sử dụng phương thức readObject() và writeObject() mặc định
- 2. Sử dụng phương thức readObject() và writeObject() riêng

(3.1) Sử dụng phương thức readObject() và writeObject() mặc định

- 1. Tạo mới một project NetBeans
 - Chon File->New Project (Ctrl+Shift+N).
 - Cửa sổ New Project xuất hiện
 - Tại pane Choose Project, chon Java trong Categories và Java Application trong Projects.
 - Click Next.
 - Tại pane Name and Location, với trường Project Name, gỗ tên project SerializeAnimationThreadNotStarted
 - Với trường Create Main Class, gõ SerializeAnimationThreadNotStarted.
 - Click Finish.



- Project **SerializeAnimationThreadNotStarted** và file **SerializeAnimationThreadNotStarted.java** xuất hiện trong editor nguồn của IDE NetBeans.
- 2. Sửa file **SerializeAnimationThreadNotStarted.java** như đoạn Code-3.11. Chú ý nhưng dòng code in đậm.

```
out.writeObject(a);
     out.close();
  } catch(IOException ex) {
     ex.printStackTrace();
  // Deserialize the object. The problem is that the
  // PersistentAnimation thread does not get started
  // automatically after the deserialization since its
  // constructor method does not get called when the
  // serialized object is deserialized. This is why
  // the PersistentAnimation class has to have its own
  // readObject() method in which the thread is
  // explicitly started.
  PersistentAnimation b = null;
  FileInputStream fis = null;
  ObjectInputStream in = null;
  try {
     fis = new FileInputStream("serializedfile");
    in = new ObjectInputStream(fis);
     b = (PersistentAnimation)in.readObject();
     in.close();
  } catch(IOException ex) {
  ex.printStackTrace();
} catch(ClassNotFoundException ex) {
     ex.printStackTrace();
  }
}
```

Code-3.11: SerializeAnimationThreadNotStarted.java

```
SerializeAnimationThreadNotStarted - NetBeans IDE 6.9
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
                                                                   Q - | Search (Ctrl+I)
                         <default config>
                                                                                     4 → ▼ 🗗
       Ser
    1 import java.io.FileInputStream;
暴
         import java.io.FileOutputStream;
    3
        import java.io.IOException;
     4
        import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
    5
     6
Projects
    7
        public class SerializeAnimationThreadNotStarted {
    8
    9 🖃
             /**
    10
              * @param args the command line arguments
    11
    12 -
             public static void main(String[] args) {
    13
                        // Create an object instance
                 PersistentAnimation a = new PersistentAnimation(1);
    15
    16
                // Serialize the object
                FileOutputStream fos = null;
    17
                ObjectOutputStream out = null;
    18
   19
                try {
    20
                    fos = new FileOutputStream("serializedfile");
    21
                    out = new ObjectOutputStream(fos);
   22
                    out.writeObject(a);
    23
                    out.close();
    24
                } catch(IOException ex) {
    <u>Q.</u>
                    ex.printStackTrace();
    26
    27
    28
                // Deserialize the object. The problem is that the
    29
                // PersistentAnimation thread does not get started
    30
                // automatically after the deserialization since its
    31
                // constructor method does not get called when the
                // serialized object is deserialized. This is why
    32
                // the PersistentAnimation class has to have its own
    33
    34
                // readObject() method in which the thread is
                 // explicitly started.
    35
屆 Output
                                                                                           INS
                                                                                      1 | 1
```

3. Tạo lớp **PersistentAnimation.java** như Code-3.12.

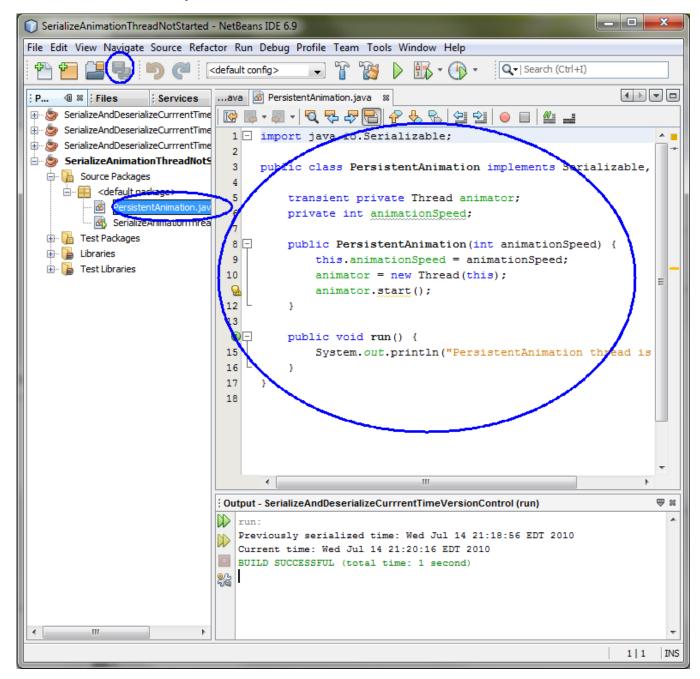
```
import java.io.Serializable;
public class PersistentAnimation implements Serializable, Runnable {
   transient private Thread animator;
   private int animationSpeed;

   public PersistentAnimation(int animationSpeed) {
      this.animationSpeed = animationSpeed;
      animator = new Thread(this);
      animator.start();
   }

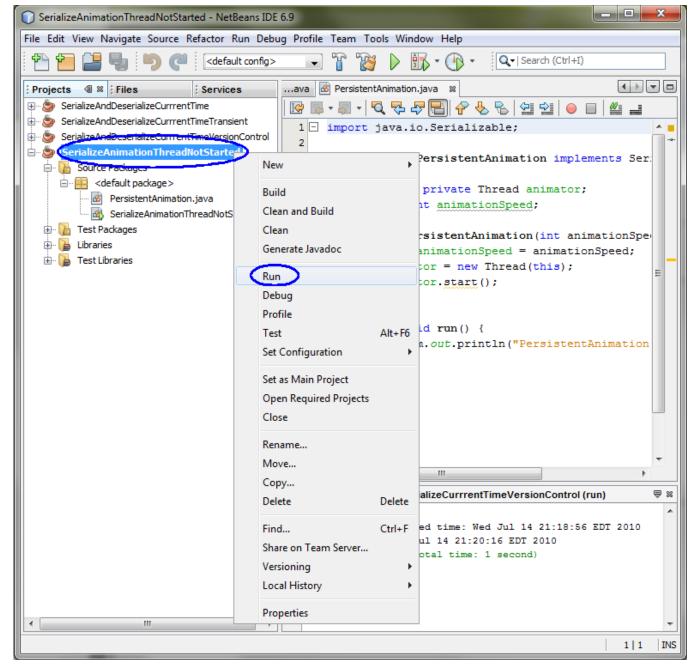
   public void run() {
      System.out.println("PersistentAnimation thread is started");
   }
```

|}

Code-3.12: PersistentAnimation.java



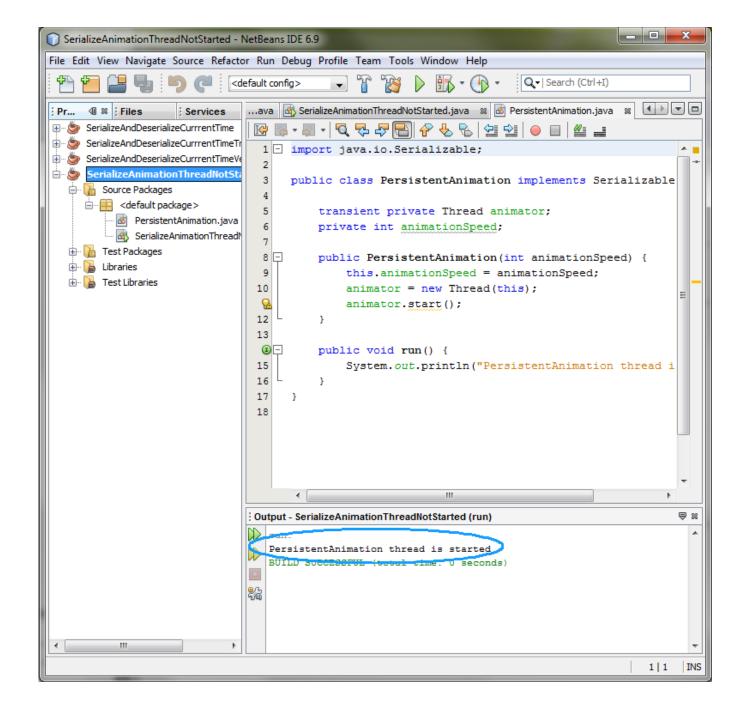
- 4. Biên dịch và chạy project
 - Click chuột phải lên project SerializeAnimationThreadNotStarted và chọn Run.



• Xem xét kết quả trong cửa sổ **Output**. (Figure-3.13) Thread không tự động bắt đầu khi đối tượng được thôi tuần tự hoá.

PersistentAnimation thread is started

Figure-3.13: Result of running SerializeAnimationThreadNotStarted application



return to top of the exercise

(3.2) Sử dụng readObject() and writeObject() riêng

1. Sửa **PersistentAnimation.java** như dưới Code-3.21. Phần sửa được in đậm màu xanh.

```
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;

public class PersistentAnimation implements Serializable, Runnable {
    transient private Thread animator;
    private int animationSpeed;

    public PersistentAnimation(int animationSpeed) {
        this.animationSpeed = animationSpeed;
        startAnimation();
    }
}
```

```
public void run() {
    System.out.println("PersistentAnimation thread is started");
}

// Provide your own writeObject method
private void writeObject(ObjectOutputStream out) throws IOException {
    out.defaultWriteObject();
}

// Provide your own readObject method
private void readObject(ObjectInputStream in) throws IOException, ClassNotFoundException {
    // our "pseudo-constructor"
    in.defaultReadObject();
    // now we are a "live" object again, so let's run rebuild and start
    startAnimation();
}

private void startAnimation() {
    animator = new Thread(this);
    animator.start();
}
```

Code-3.21: PersistentAnimation.java

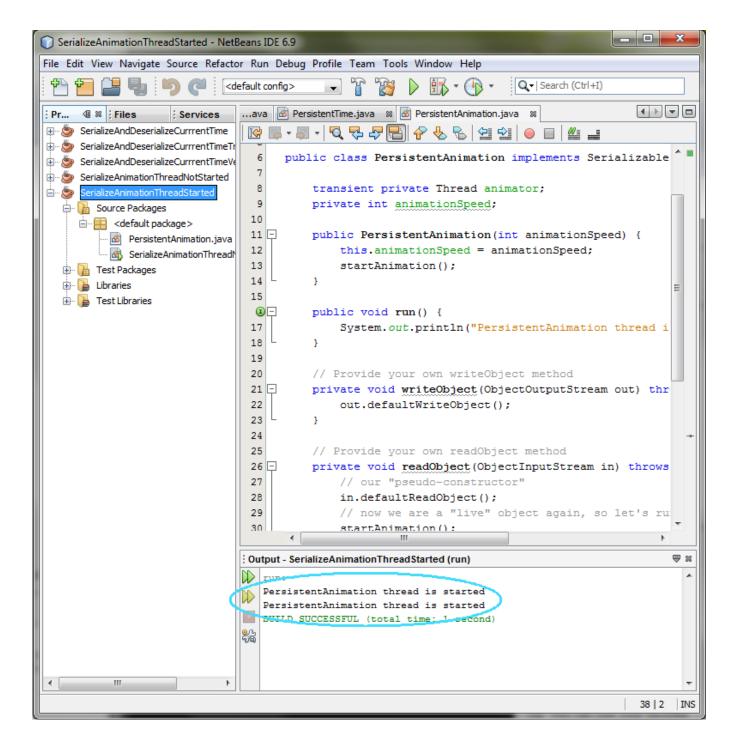
```
SerializeAnimationThreadStarted - NetBeans IDE 6.9
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
                                                                        Q - | Search (Ctrl+I)
                            <default config>
                                                                                           4 → ▼ 🗗
        j SQL Command 46 № B PersistentTime.java № B PersistentTime.java № B PersistentAnimation.java №
         Ser
듛
     6
         public class PersistentAnimation implements Serializable, Runnable {
     7
     8
              transient private Thread animator;
9
              private int animationSpeed;
    10
Projects
    11 -
              public PersistentAnimation(int animationSpeed) {
    12
                  this animationSpeed = animationSpeed;
4
    13
                  startAnimation();
    14
    15
     (I)
              public void run() {
    17
                  System.out.println("PersistentAnimation thread is started");
    18
    19
    20
                    vide your own writeObject method
    21 🖃
              rivate void writeObject(ObjectOutputStream out) throws IOExceptIon
    22
                  out.defaultWriteObject();
    23
    24
    25
              // Provide your own readObject method
              private void readObject(ObjectInputStream in) throws IOException, ClassNotFo
    26 -
                  // our "pseudo-constructor"
    27
                  in.defaultReadObject();
    28
    29
                  // now we are a "live" object again, so let's run rebuild and start
    30
                  startAnimation();
    31
    33
    34 🖃
              private void startAnimation() {
    35
                  animator = new Thread(this);
                  animator.start();
    36
    37
    38
Output
                                                                                            38 | 2 INS
```

2. Biên dịch và chạy project

- Click chuôt phải vào SerializeAnimationThreadStarted và chon Run.
- Xem xét kết quả xuất hiện trong cửa sổ Output. Thread tự động bắt đầu khi đối tượng được thôi tuần tự hoá.

PersistentAnimation thread is started
PersistentAnimation thread is started

Figure-3.22: Result of running SerializeAnimationThreadStarted application



return to top of the exercise

Summary

Trong bài lab này, chúng ta đã học cách sử dụng và tạo phương thức readObject() và writeObject() riêng.

return to the top

Luyện tập

- 1. Tạo project SerializeTime:
 - Viết lớp MyClassToBePersisted.java, với các trường:
 String name

- $\circ \ \text{int age}$
- String hobby
- String nameOfSchool
- int yearStarted (transient)
- Viết lớp SerializeMyClassToBePersisted.java, để tạo một thực thể của lớp MyClassToBePersisted và tuần tự hoá nó trong phương thức main.

 • Viết lớp DeserializeMyClassToBePersisted.java, để đọc file ở mục trước và thôi tuần tự hoá thực thể của lớp
- MyClassToBePersisted trong phương thức main.
- Sửa lớp MyClassToBePersisted.java, để chứa version control.