Java Stream I/O

Môn học: Lập trình mạng
Giảng viên: TS. Nguyễn Trọng Khánh

Resources

- **Basic I/O tutorial** from java.sun.com

Exercises

Exercise 1: Byte stream

Trong bài lab này, bạn sẽ làm quen với lớp FileInputStream thuộc luồng InputStream và FileOutputStream thuộc luồng OutputStream, để đọc và ghi file.

(1.1) Viết ứng dụng sử dụng FileInputStream và FileOutputStream

0. Khởi động NetBeans IDE.
1. Tạo mới một project NetBeans

- Chọn **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**, gõ tên project **FileInputOutputStream**.
- Với trường **Create Main Class**, gõ **FileInputOutputStream**. (Figure-1.10)
- Click **Finish**.

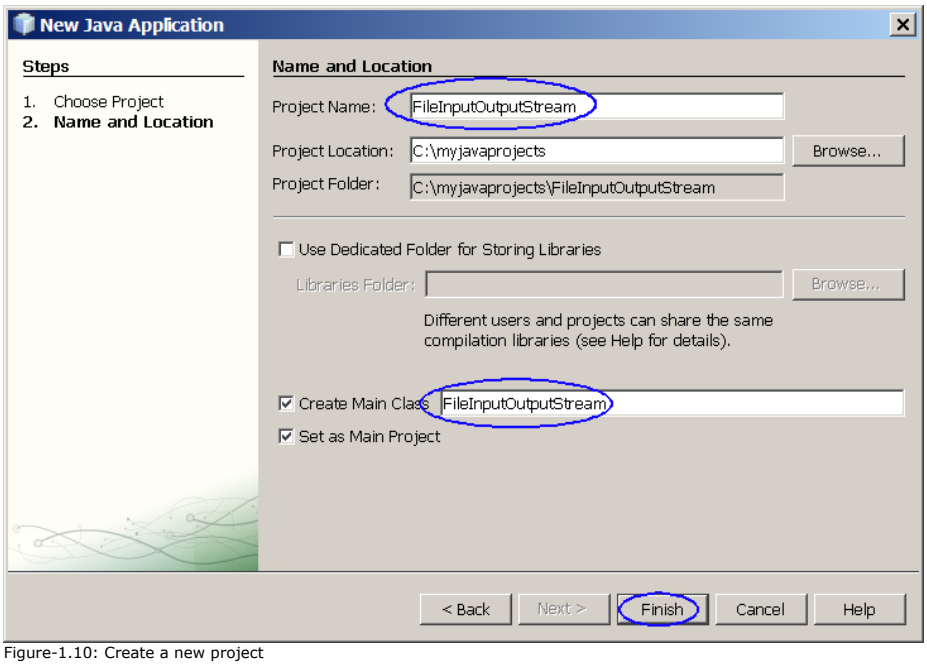


Figure-1.10: Create a new project

- Project **FileInputOutputStream** và file **FileInputOutputStream.java** xuất hiện trong IDE NetBeans.

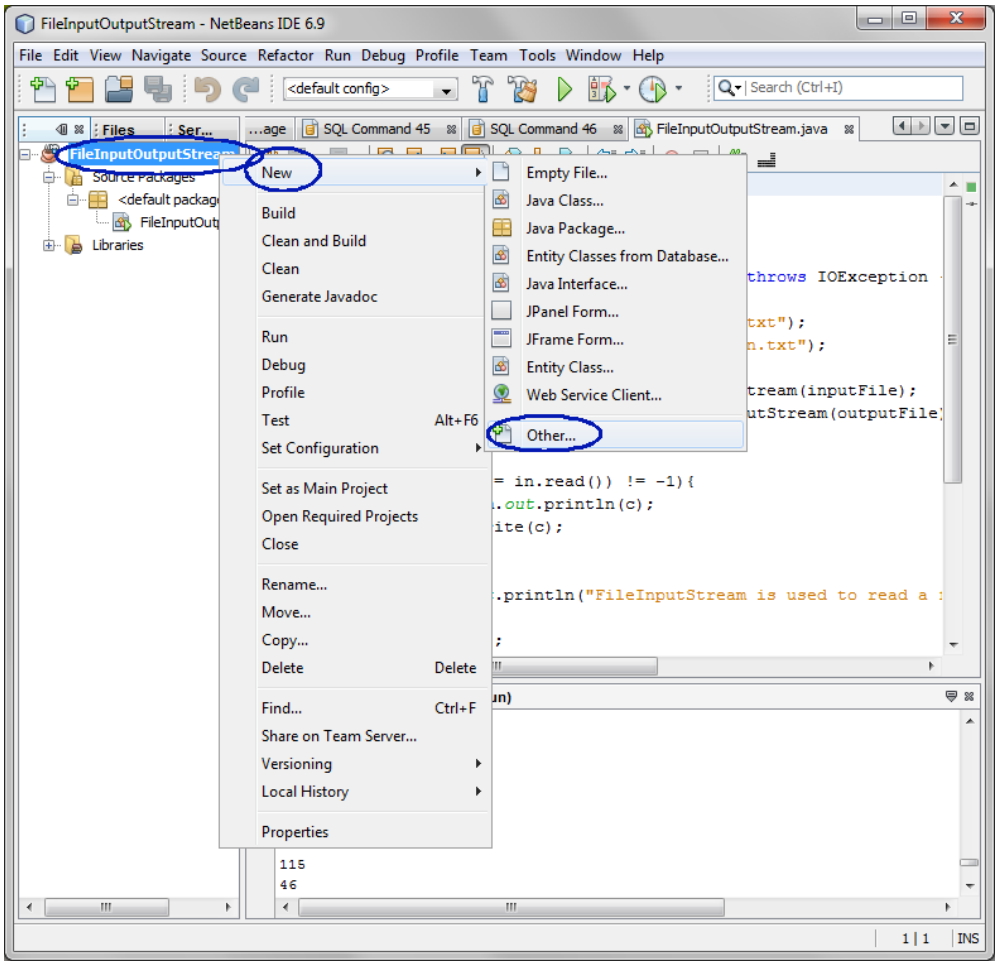
2. Sửa file **FileInputOutputStream.java** như đoạn Code-1.11. Chú ý nhưng dòng code in đậm.

```
import java.io.*;

public class FileInputOutputStream {

    public static void main(String[] args) throws IOException {

        File inputFile = new File("farrago.txt");
        File outputFile = new File("outagain.txt");

        FileInputStream in = new FileInputStream(inputFile);
        FileOutputStream out = new FileOutputStream(outputFile);
        int c;

        while ((c = in.read()) != -1){
            System.out.println(c);
            out.write(c);
        }

        System.out.println("FileInputStream is used to read a file and FileOutPutStream is used for writing.");

        in.close();
        out.close();
    }
}
```

Code-1.11: FileInputOutputStream.java

3. Tạo file input **farrago.txt**.

- Click chuột phải vào project **FileInputOutputStream** và chọn **New->Other**.



- Cửa sổ **New File** xuất hiện.
- Chon **Other** trong **Categories** và **Empty File** trong **File Types**. (Figure-1.12)
- Click Next.

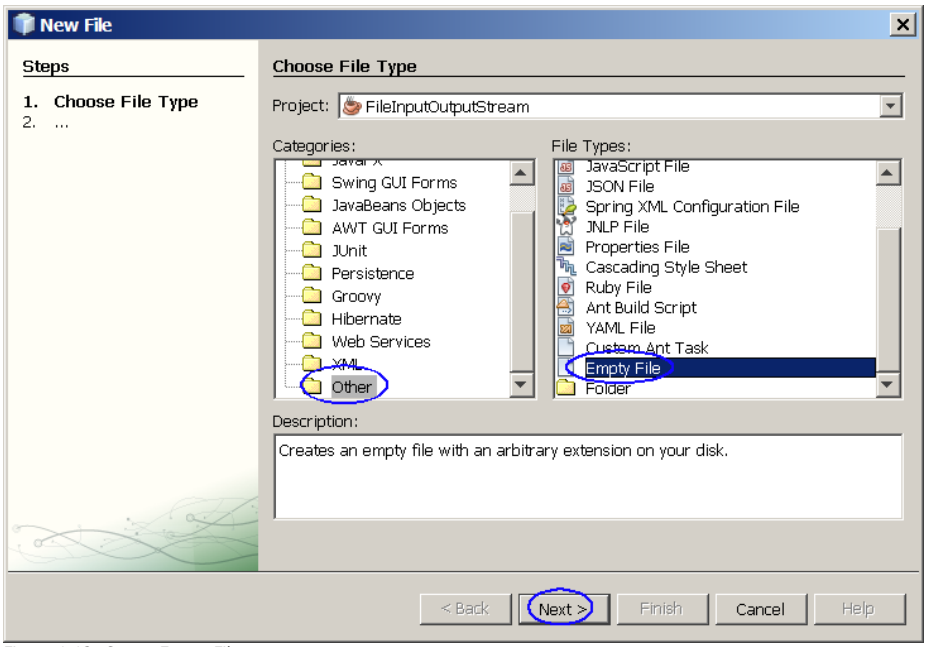
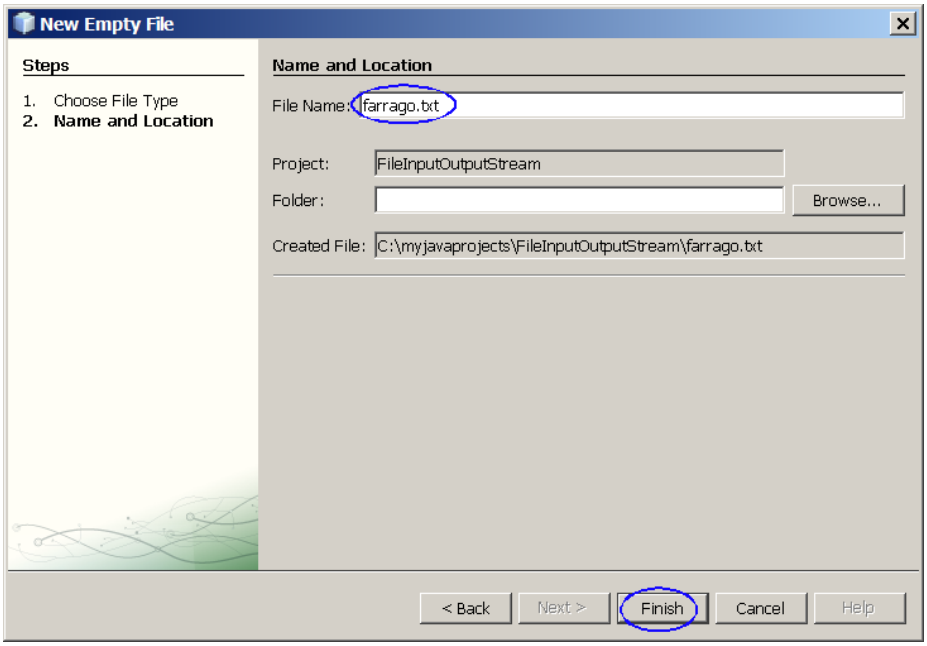


Figure-1.12: Create Empty File

- Cửa sổ **New Empty File** xuất hiện.
- Trong trường **File Name**, gõ **farrago.txt**.
- Click Finsh.



- File **farrago.txt** xuất hiện trong của sổ editor.
- Cut và paste nội dung ở dưới InputFile-1.14 vào file.

```
So she went into the garden to cut a cabbage-leaf, to
make an apple-pie; and at the same time a great
she-bear, coming up the street, pops its head into the
shop. 'What! no soap?' So he died, and she very
imprudently married the barber; and there were
present the Picninnies, and the Joblillies, and the
Garyalies, and the grand Panjandrum himself, with the
little round button at top, and they all fell to playing
the game of catch as catch can, till the gun powder ran
out at the heels of their boots.

Samuel Foote 1720-1777
```

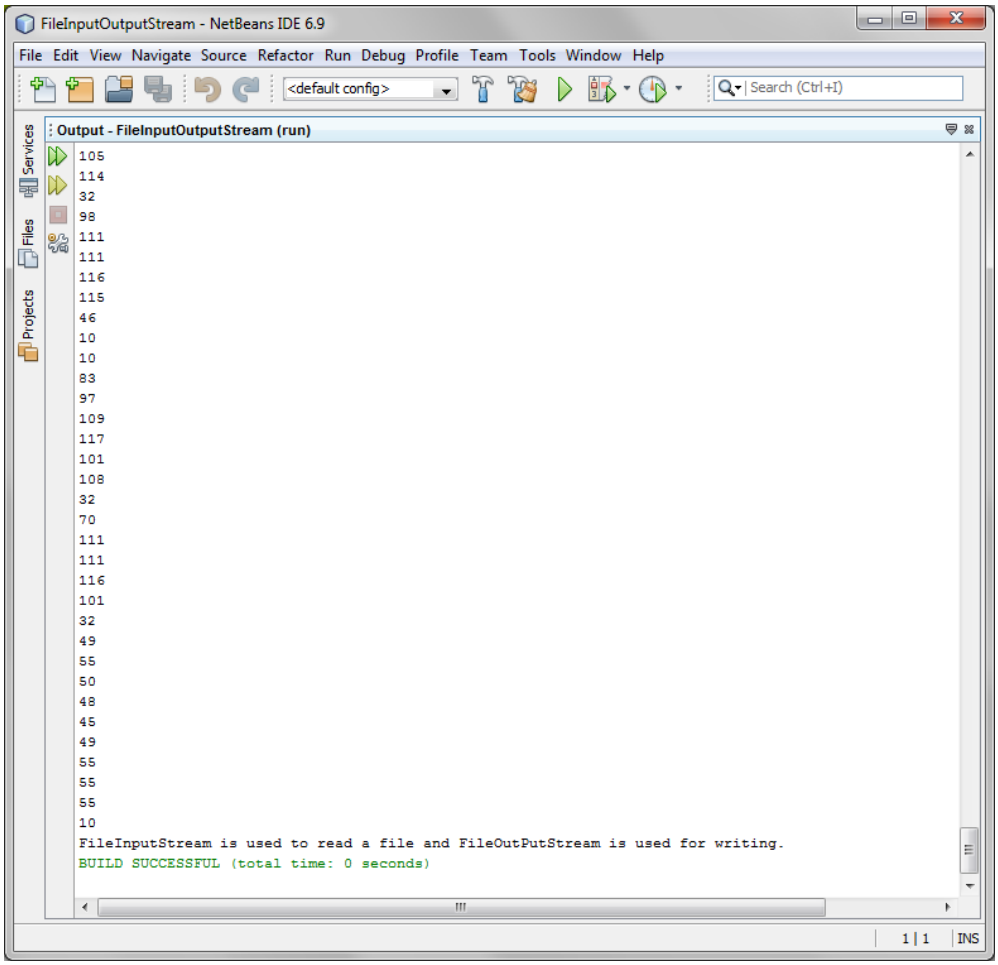
InputFile-1.14: farrago.txt

4. Dịch và chạy project

- Click Chuột phải vào project **FileInputOutputStream** và chọn **Run**.
- Xem kết quả xuất hiện trong cửa sổ **Output**. (Figure-1.15)

```
...
55
55
55
10
FileInputStream is used to read a file and FileOutPutStream is used for writing.
```

Figure-1.15: Kết quả chạy ứng dụng FileInputOutputStream



5. **Thực hành**: Sửa **FileInputOutputStream.java** để đọc file myowninputfile.txt và ghi nội dung ra file myownoutputfile.txt

return to top of the exercise

Summary

Cách sử dụng lớp FileInputStream thuộc kiểu InputStream và lớp FileOutputStream thuộc kiểu OutputStream để đọc và ghi dữ liệu ra file.

return to the top

Exercise 2: Character Stream

Trong bài thực hành này, chúng ta sẽ làm quen với lớp FileReader thuộc kiểu Reader và lớp FileWriter thuộc kiểu Writer để đọc và ghi file.

(2.1) Viết ứng dụng sử dụng FileReader và FileWriter

1. Tạo mới project NetBeans

- Chọn **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**, gõ tên project **FileReaderWriter**
- Với trường **Create Main Class**, gõ **FileReaderWriter**.
- Click **Finish**.

- Project **FileReaderWriter** và file **FileReaderWriter.java** xuất hiện trong IDE NetBeans.

2. Sửa file **FileReaderWriter.java** như đoạn Code-2.11. Chú ý nhưng dòng code in đậm.

```
import java.io.*;

public class FileReaderWriter {

    public static void main(String[] args) throws IOException {

        File inputFile = new File("farrago.txt");
        File outputFile = new File("outagain.txt");

        FileReader in = new FileReader(inputFile);
        FileWriter out = new FileWriter(outputFile);
        int c;

        while ((c = in.read()) != -1){
            System.out.println(c);
            out.write(c);
        }

        System.out.println("FileReader is used to read a file and FileWriter is used for writing.");

        in.close();
        out.close();
```

```
    }
}
```
Code-2.11: FileReaderWriter.java

3. Tạo file input **farrago.txt**.

```
So she went into the garden to cut a cabbage-leaf, to
make an apple-pie; and at the same time a great
she-bear, coming up the street, pops its head into the
shop. 'What! no soap?' So he died, and she very
imprudently married the barber; and there were
present the Picninnies, and the Joblillies, and the
Garyalies, and the grand Panjandrum himself, with the
little round button at top, and they all fell to playing
the game of catch as catch can, till the gun powder ran
out at the heels of their boots.

Samuel Foote 1720-1777
```
File-2.12: farrago.txt

4. Dịch và chạy project

- Click Chuột phải vào project **FileReaderWriter** và chọn **Run**.
- Xem kết quả xuất hiện trong cửa sổ Output. (Figure-2.13)

```
...
55
55
55
10
FileReader is used to read a file and FileWriter is used for writing.
```
Figure-2.13: Result of running FileReaderWriter application

5. **Thực hành**: Sửa **FileReaderWriter.java** để đọc file myowninputfile.txt và ghi nội dung ra file myownoutputfile.txt

return to top of the exercise

Summary

Cách sử dụng lớp FileReader thuộc kiểu Reader và lớp FileWriter thuộc kiểu Writer để đọc và ghi dữ liệu ra file.

return to the top

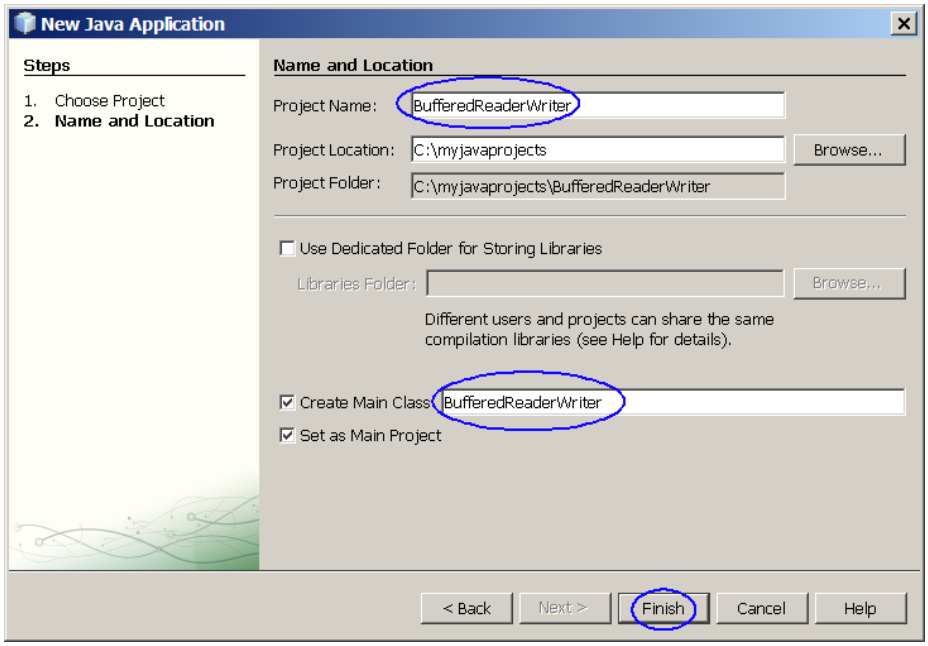
Exercise 3: Buffered Stream

Trong bài lab này, bạn sẽ làm quen với lớp BufferedReader và BufferedWriter để thao tác vào ra với buffer

(3.1) Viết ứng dụng sử dụng BufferedReader và BufferedWriter

1. Create a new NetBeans project

- Tạo mới một project NetBeans
- Chọn **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**, gõ tên project **BufferedReaderWriter**.
- Với trường **Create Main Class**, gõ **BufferedReaderWriter**.
- Click **Finish**.



- Project **BufferedReaderWriter** và file **BufferedReaderWriter.java** xuất hiện trong IDE NetBeans.

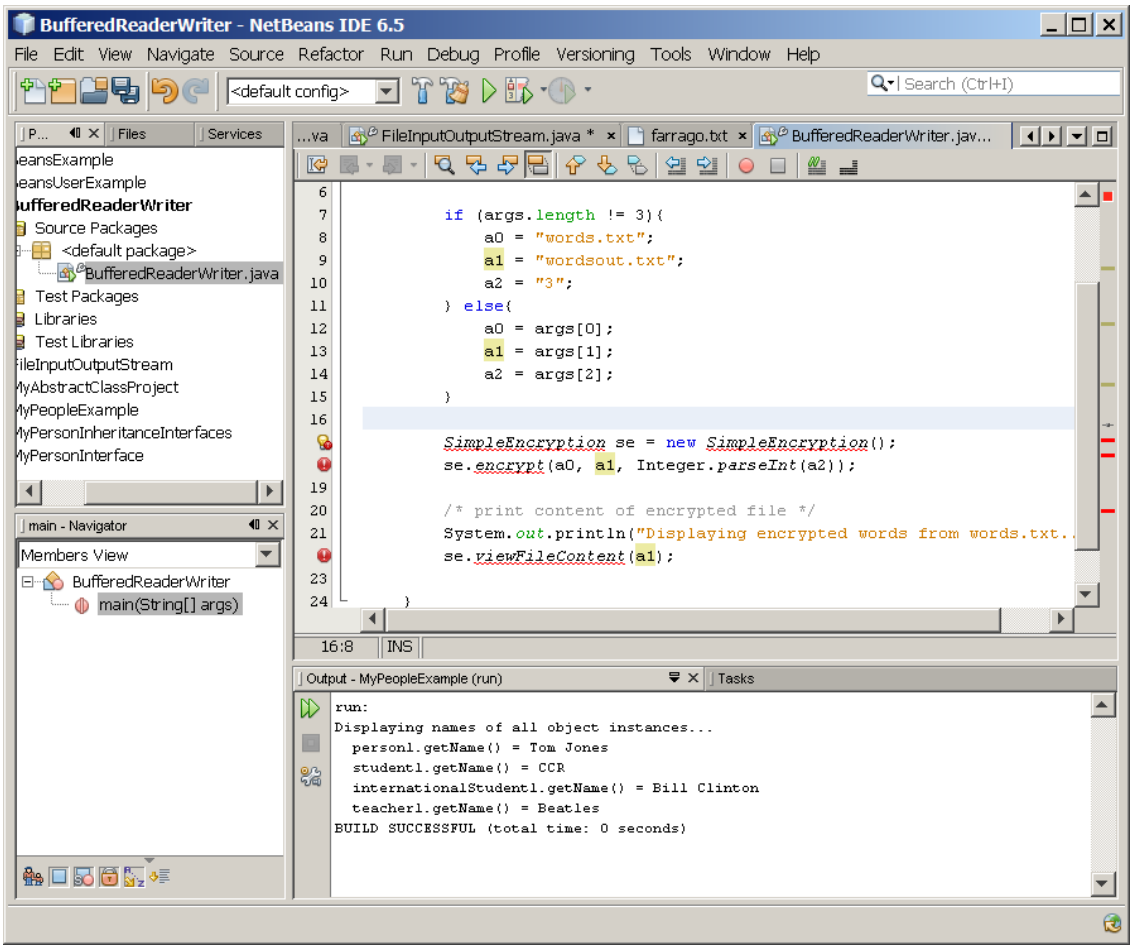
2. Sửa file **BufferedReaderWriter.java** như đoạn Code-3.11. Chú ý nhưng dòng code in đậm.

```
public class BufferedReaderWriter {

   public static void main(String args[]) {

      String a0, a1, a2;
```

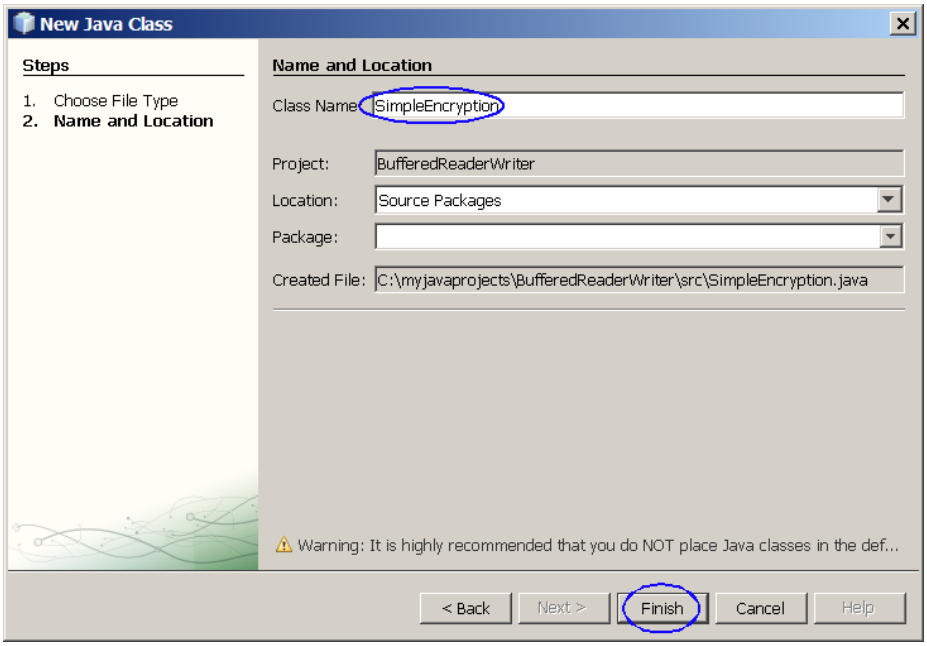
```
      if (args.length != 3){
         a0 = "words.txt";
         a1 = "wordsout.txt";
         a2 = "3";
      } else{
         a0 = args[0];
         a1 = args[1];
         a2 = args[2];
      }

      SimpleEncryption se = new SimpleEncryption();
      se.encrypt(a0, a1, Integer.parseInt(a2));

      /* print content of encrypted file */
      System.out.println("Displaying encrypted words from words.txt...");
      se.viewFileContent(a1);

   }
}
```

Code-3.11: BufferedReaderWriter.java

- Chú ý có một số lỗi xuất hiện.



3. Viết lớp **SimpleEncryption.java** như Code-3.12. Nghiên cứu những dòng code in đậm.

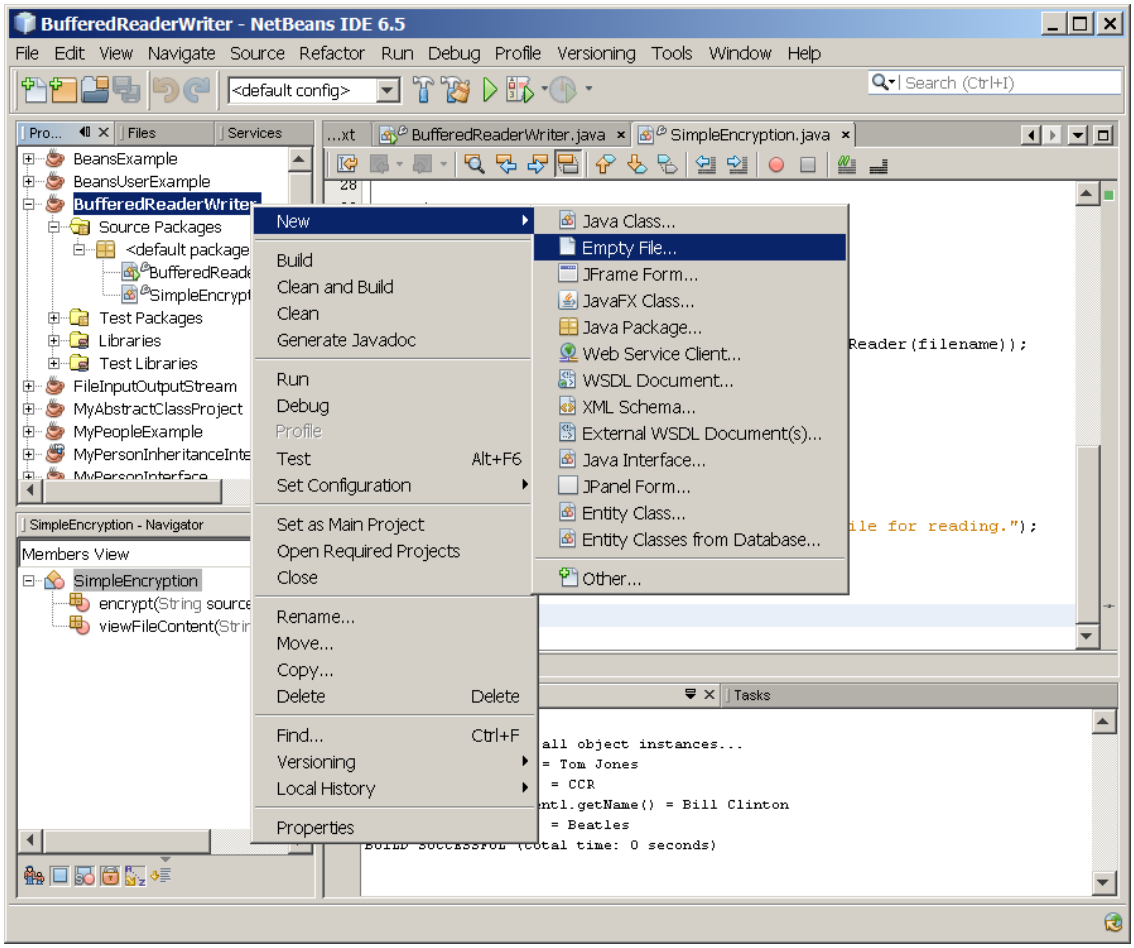


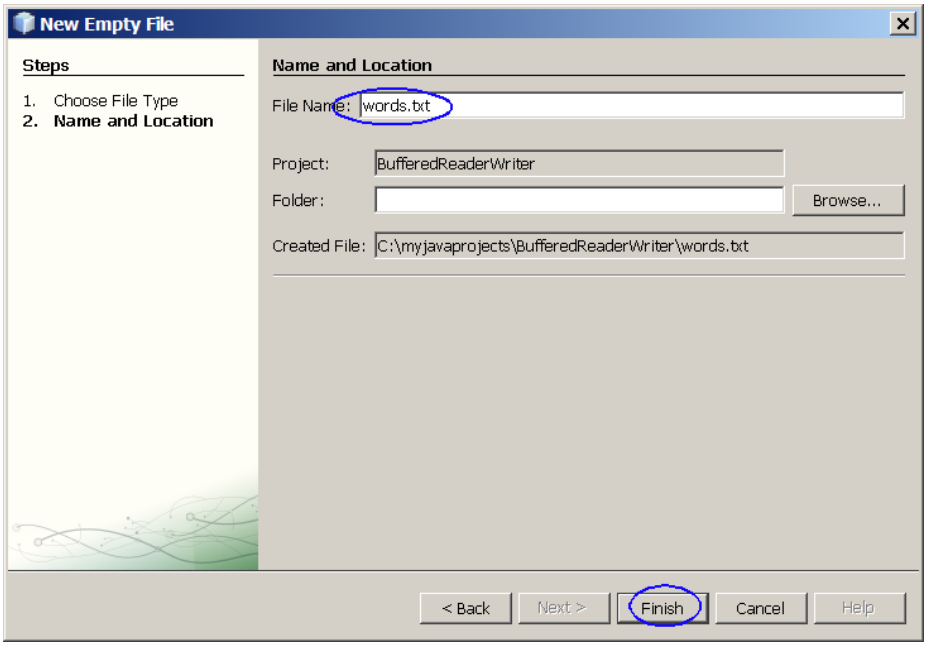
- Sửa code như sau.

```
import java.io.*;

class SimpleEncryption {

  void encrypt(String source, String dest, int shiftSize) {

    BufferedReader reader;
    BufferedWriter writer;

    try {
      reader = new BufferedReader(new FileReader(source));
      writer = new BufferedWriter(new FileWriter(dest));
      String line = reader.readLine();
      int data;
      while (line != null) {
        for (int i = 0; i < line.length(); i++) {
          data =(int) line.charAt(i) + shiftSize;
          writer.write(data);
        }
        writer.write((int)'\n');
        line = reader.readLine();
      }
      reader.close();
      writer.close();
    } catch (IOException ie) {
      System.out.println("Failed encrypting the file content.");
    }

  }

  void viewFileContent(String filename) {

    BufferedReader reader;
    try {
      reader = new BufferedReader(new FileReader(filename));
      String line = reader.readLine();
      while (line != null) {
        System.out.println(line);
        line = reader.readLine();
      }
      reader.close();
    } catch (IOException ie) {
      System.out.println("Failed to open file for reading.");
    }
  }

}
```

Code-3.12: SimpleEncryption.java

4. Tạo **words.txt**. (File-3.13)



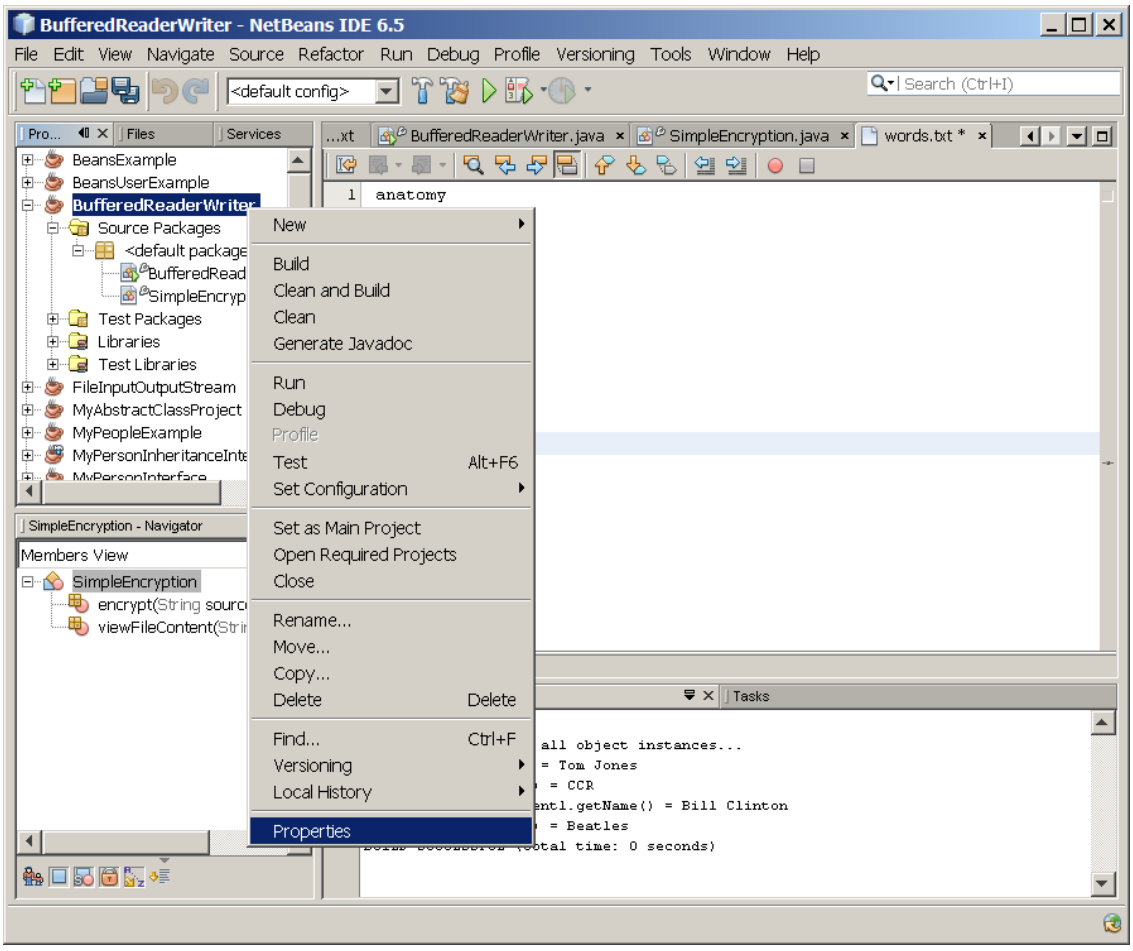



```
anatomy
animation
applet
application
argument
bolts
class
communicate
string
threads
tools
user
```

File-3.13: words.txt

5. Tạo đối số để truyền theo dòng lệnh.

- Click chuột phải lên project **BufferedReaderWriter** chọn **Properties**.



- Cửa sổ **Project Properties** xuất hiện.
- Chọn **Run**, trong trường **Arguments**, gõ **words.txt wordsout.txt 3**. (Figure-3.14)

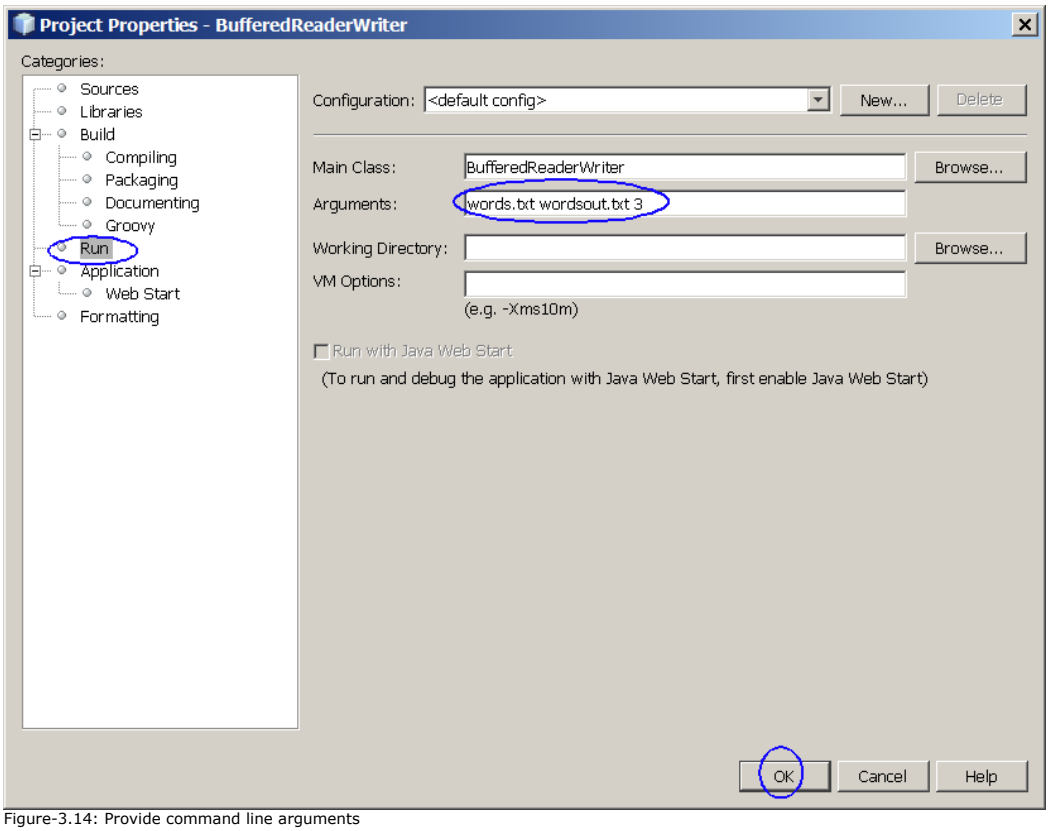


Figure-3.14: Provide command line arguments

6.Dịch và chạy project

- Click Chuột phải vào project **BufferedReaderWriter** và chọn **Run**.
- Xem kết quả xuất hiện trong cửa sổ Output.(Figure-3.15)

```
Displaying encrypted words from words.txt...
dqdwrp|
dqlpdwlrq
dssohw
dssolfdwlrq
dujxphqw
erowv
fodvv
frppxqlfdwh
vwulqj
wkuhdgv
wrrov
xvhu
```
Figure-3.15: Result of running BufferedReaderWriter application

return to top of the exercise

Summary

Trong bài lab này, bạn đã làm quen với lớp BufferedReader và BufferedWriter để thao tác vào ra với buffer.

return to the top

Exercise 4: Data Stream

Trong bài lab này, bạn sẽ làm quen với lớp DataInputStream và DataOutputStream, để đọc và ghi các kiểu dữ liệu gốc.

1. **DataInputStream and DataOutputStream**
2. **DataInput and DataOut**

(4.1) DataInputStream and DataOutputStream

1. Tạo mới một project NetBeans

- Chọn 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, gõ tên project **DataInputOutputStream**.
- Với trường Create Main Class, gõ **DataInputOutputStream**.
- Click Finish.

- Project **DataInputOutputStream** và file **DataInputOutputStream.java** xuất hiện trong IDE NetBeans.

2. Sửa file **DataInputOutputStream.java** như đoạn Code-4.11. Chú ý nhưng dòng code in đậm.

```
import java.io.*;

public class DataInputOutputStream {

    public static void main(String[] args) throws IOException {
```

```
        // write the data out
        DataOutputStream out = new DataOutputStream(new
              FileOutputStream("invoice"));

        double[] prices = { 19.99, 9.99, 15.99, 3.99, 4.99 };
        int[] units = { 12, 8, 13, 29, 50 };
        String[] descs = { "Java T-shirt",
        "Java Mug",
        "Duke Juggling Dolls",
        "Java Pin",
        "Java Key Chain" };

        for (int i = 0; i < prices.length; i ++) {
           out.writeDouble(prices[i]);
           out.writeChar('\t');
           out.writeInt(units[i]);
           out.writeChar('\t');
           out.writeChars(descs[i]);
           out.writeChar('\n');
        }
        out.close();

        // read it in again
        DataInputStream in = new DataInputStream(new
              FileInputStream("invoice"));

        double price;
        int unit;
        StringBuffer desc;
        double total = 0.0;

        String lineSepString = System.getProperty("line.separator");
        char lineSep = lineSepString.charAt(lineSepString.length()-1);

        try {
           while (true) {
              price = in.readDouble();
              in.readChar();       // throws out the tab
              unit = in.readInt();
              in.readChar();       // throws out the tab
              char chr;
              desc = new StringBuffer(20);
              while ((chr = in.readChar()) != lineSep)
                 desc.append(chr);
              System.out.println("You've ordered " +
                    unit + " units of " +
                    desc + " at $" + price);
              total = total + unit * price;
           }
        } catch (EOFException e) {
        }
        System.out.println("For a TOTAL of: $" + total);
        in.close();
     }
}
```

Code-4.11: DataInputOutputStream.java

5 Dịch và chạy project

- Click Chuột phải vào project **DataInputOutputStream** và chọn Run.
- Xem kết quả xuất hiện trong cửa sổ Output. (Figure-4.14 below)

```
You've ordered 12 units of Java T-shirt at $19.99
You've ordered 8 units of Java Mug at $9.99
You've ordered 13 units of Duke Juggling Dolls at $15.99
You've ordered 29 units of Java Pin at $3.99
You've ordered 50 units of Java Key Chain at $4.99
For a TOTAL of: $892.8800000000001
```

Figure-4.14: Result of running DataInputOutputStream application

return to top of the exercise

(4.2) DataInput and DataOutput

Trong phần này, chúng ta sẽ luyện tập xử lý luồng thực hiện các thao tác kiểm tra dữ liệu.

1. Tạo mới một project NetBeans

- Chọn 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, gõ tên project **CustomDataInputOutput**.
- Với trường Create Main Class, gõ **CustomDataInputOutput**.
- Click **Finish**.

- Project **CustomDataInputOutput** và file **CustomDataInputOutput.java** xuất hiện trong IDE NetBeans.

2. Sửa file **CustomDataInputOutput.java** như đoạn Code-4.21. Chú ý nhưng dòng code in đậm.

```
import java.io.*;

public class CustomDataInputOutput {
   public static void main(String[] args) throws IOException {

      Adler32 inChecker = new Adler32();
      Adler32 outChecker = new Adler32();
      CheckedDataInput in = null;
      CheckedDataOutput out = null;

      try {
         in = new CheckedDataInput(
            new DataInputStream(
                  new FileInputStream("farrago.txt")),
            inChecker);
         out = new CheckedDataOutput(
             new DataOutputStream(
                   new FileOutputStream("outagain.txt")),
             outChecker);
      } catch (FileNotFoundException e) {
         System.err.println("CheckedIOTest: " + e);
```

```
            System.exit(-1);
        } catch (IOException e) {
            System.err.println("CheckedIOTest: " + e);
            System.exit(-1);
        }

        boolean EOF = false;

        while (!EOF) {
            try {
                int c = in.readByte();
                out.write(c);
            } catch (EOFException e) {
                EOF = true;
            }
        }

        System.out.println("Input stream check sum: " +
                in.getChecksum().getValue());
        System.out.println("Output stream check sum: " +
                out.getChecksum().getValue());
    }
}
```

Code-4.21: CustomDataInputOutput.java

3. Tạo file **CheckedDataInput.java**. (Code-4.22)

```
import java.io.*;

public class CheckedDataInput {

    private Checksum cksum;
    private DataInput in;

    public CheckedDataInput(DataInput in, Checksum cksum) {
        this.cksum = cksum;
        this.in = in;
    }

    public byte readByte() throws IOException {
        byte b = in.readByte();
        cksum.update(b);
        return b;
    }

    public void readFully(byte[] b) throws IOException {
        in.readFully(b, 0, b.length);
        cksum.update(b, 0, b.length);
    }

    public void readFully(byte[] b, int off, int len) throws IOException {
        in.readFully(b, off, len);
        cksum.update(b, off, len);
    }

    public Checksum getChecksum() {
        return cksum;
    }
}
```

Code-4.22: CheckedDataInput.java

4. Tạo **CheckedDataOutput.java**. (Code-4.23)

```
import java.io.*;

public class CheckedDataOutput {

    private Checksum cksum;
    private DataOutput out;

    public CheckedDataOutput(DataOutput out, Checksum cksum) {
        this.cksum = cksum;
        this.out = out;
    }

    public void write(int b) throws IOException {
        out.write(b);
        cksum.update(b);
    }

    public void write(byte[] b) throws IOException {
        out.write(b, 0, b.length);
        cksum.update(b, 0, b.length);
    }

    public void write(byte[] b, int off, int len) throws IOException {
        out.write(b, off, len);
        cksum.update(b, off, len);
    }

    public Checksum getChecksum() {
        return cksum;
    }
}
```

Code-4.23: CheckedDataOutput.java

5. Tạo **Checksum.java**. (Code-4.24)

```
public interface Checksum {
    /**
     * Updates the current checksum with the specified byte.
     */
    public void update(int b);

    /**
     * Updates the current checksum with the specified array of bytes.
     */
    public void update(byte[] b, int off, int len);
```

```
    /**
     * Returns the current checksum value.
     */
    public long getValue();

    /**
     * Resets the checksum to its initial value.
     */
    public void reset();
}
```

Code-4.24: Checksum.java

6. Tạo **Adler32.java**. (Code-4.25)

```
public class Adler32 implements Checksum {
    private int value = 1;

    /*
     * BASE is the largest prime number smaller than 65536
     * NMAX is the largest n such that 255n(n+1)/2 + (n+1)(BASE-1) <= 2^32-1
     */
    private static final int BASE = 65521;
    private static final int NMAX = 5552;

    /**
     * Update current Adler-32 checksum given the specified byte.
     */
    public void update(int b) {
        int s1 = value & 0xffff;
        int s2 = (value >> 16) & 0xffff;
        s1 += b & 0xff;
        s2 += s1;
        value = ((s2 % BASE) << 16) | (s1 % BASE);
    }

    /**
     * Update current Adler-32 checksum given the specified byte array.
     */
    public void update(byte[] b, int off, int len) {
        int s1 = value & 0xffff;
        int s2 = (value >> 16) & 0xffff;

        while (len > 0) {
            int k = len < NMAX ? len : NMAX;
            len -= k;
            while (k-- > 0) {
                s1 += b[off++] & 0xff;
                s2 += s1;
            }
            s1 %= BASE;
            s2 %= BASE;
        }
        value = (s2 << 16) | s1;
    }

    /**
     * Reset Adler-32 checksum to initial value.
     */
    public void reset() {
        value = 1;
    }

    /**
     * Returns current checksum value.
     */
    public long getValue() {
        return (long)value & 0xffffffff;
    }
}
```

Code-4.25: Adler32.java

7. Tạo file input **farrago.txt**.

```
So she went into the garden to cut a cabbage-leaf, to
make an apple-pie; and at the same time a great
she-bear, coming up the street, pops its head into the
shop. 'What! no soap?' So he died, and she very
imprudently married the barber; and there were
present the Picninnies, and the Joblillies, and the
Garyalies, and the grand Panjandrum himself, with the
little round button at top, and they all fell to playing
the game of catch as catch can, till the gun powder ran
out at the heels of their boots.

Samuel Foote 1720-1777
```

File-2.12: farrago.txt

8. Dịch và chạy project

- Click Chuột phải vào project **CustomDataInputOutput** và chọn Run.
- Xem kết quả xuất hiện trong cửa sổ Output.Build and run the project. (Figure-4.14)

```
Input stream check sum: 736868089
Output stream check sum: 736868089
```

Figure-4.14: Result

return to top of the exercise

Summary

Trong bài lab này, bạn đã làm quen với lớp DataInputStream và DataOutputStream, để đọc và ghi các kiểu dữ liệu gốc.

return to the top

Exercise 5: Object Stream

Trong bài lab này, bạn sẽ làm quen với lớp ObjectInputStream và ObjectOutputStream, để đọc và ghi các đối tượng.

(5.1) Viết ứng dụng sử dụng ObjectInputStream và ObjectOutputStream

1. Tạo mới một project NetBeans

- Chọn 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, gõ tên project **ObjectInputOutputStream**.
- Với trường Create Main Class, gõ **ObjectInputOutputStream**.
- Click **Finish**.

- Project **ObjectInputOutputStream** và file **ObjectInputOutputStream.java** xuất hiện trong IDE NetBeans.

2. Sửa file DataInputOutputStream.java như đoạn Code-5.11. Chú ý nhưng dòng code in đậm.

```
import java.io.*;

public class ObjectInputOutputStream {

   public static void main(String[] args) {

      Card3 card = new Card3(12, Card3.SPADES);
      System.out.println("Card to write is: " + card);

      try {
         FileOutputStream out = new FileOutputStream("card.out");
         ObjectOutputStream oos = new ObjectOutputStream(out);
         oos.writeObject(card);
         oos.flush();
      } catch (Exception e) {
         System.out.println("Problem serializing: " + e);
      }

      try {
         FileInputStream in = new FileInputStream("card.out");
         ObjectInputStream ois = new ObjectInputStream(in);
         card = (Card3)(ois.readObject());
      } catch (Exception e) {
         System.out.println("Problem serializing: " + e);
      }

      System.out.println("Card read is: " + card);
   }
}
```

Code-5.11: ObjectInputOutputStream.java

3. Tạo lớp **Card3.java** như dưới Code-5.12. Nghiên cứu phần in đậm

```
import java.io.Serializable;

public class Card3 implements Serializable {
   private int suit = UNASSIGNED;
   private int number = UNASSIGNED;

   public final static int UNASSIGNED = -1;

   public final static int DIAMONDS = 1;
   public final static int CLUBS = 2;
   public final static int HEARTS = 3;
   public final static int SPADES = 4;

   public final static int ACE = 1;
   public final static int KING = 13;

   public Card3(int number, int suit) {
      if (isValidNumber(number)) {
         this.number = number;
      } else {
         //Error
      }

      if (isValidSuit(suit)) {
         this.suit = suit;
      } else {
         //Error
      }
   }

   public int getSuit() {
      return suit;
   }

   public int getNumber() {
      return number;
   }

   public static boolean isValidNumber(int number) {
      if (number >= ACE && number <= KING) {
         return true;
      } else {
         return false;
      }
   }
   public static boolean isValidSuit(int suit) {
      if (suit >= DIAMONDS && suit <= SPADES) {
         return true;
      } else {
         return false;
      }
   }

   public boolean equals(Object obj) {
      if (obj instanceof Card3) {
      Card3 card = (Card3)obj;
         if (card.getNumber() == this.number && card.getSuit() == this.suit) {
```

```
            return true;
         } else {
            return false;
         }
      } else {
         return false;
      }
   }
   public int hashCode() {
      return number * suit;
   }
   public String toString() {
      return numberToString(this.number) + " of "
           + suitToString(this.suit);
   }

   public static String numberToString(int number) {
      String result = "";
      switch (number) {
         case ACE: result =  "Ace"; break;
         case 2: result = "Two"; break;
         case 3: result = "Three"; break;
         case 4: result = "Four"; break;
         case 5: result = "Five"; break;
         case 6: result = "Six"; break;
         case 7: result = "Seven"; break;
         case 8: result = "Eight"; break;
         case 9: result = "Nine"; break;
         case 10: result = "Ten"; break;
         case 11: result = "Jack"; break;
         case 12: result = "Queen"; break;
         case KING: result = "King"; break;
         case UNASSIGNED: result = "Invalid Number"; break;
      }
      return result;
   }

   public static String suitToString(int suit) {
      String result = "";
      switch (suit) {
         case DIAMONDS: result = "Diamonds"; break;
         case CLUBS: result = "Clubs"; break;
         case HEARTS: result = "Hearts"; break;
         case SPADES: result = "Spades"; break;
         case UNASSIGNED: result = "Invalid Suit"; break;
      }
      return result;
   }
}
```

Code-5.12: Card3.java

4. Dịch và chạy project

- Click Chuột phải vào project **ObjectInputOutputStream** và chọn Run.
- Xem kết quả xuất hiện trong cửa sổ Output.

```
Card to write is: Queen of Spades
Card read is: Queen of Spades
```

Figure-5.15: Result

return to top of the exercise

Summary

Trong bài lab này, bạn đã được làm quen với lớp ObjectInputStream và ObjectOutputStream, để đọc và ghi các đối tượng.

return to the top

Exercise 6: File and Directory Handling

Trong bài lab này, bạn sẽ làm quen với lớp File để thực hiện một số thao tác liên quan đến file và thư mục.

1. **File handling**
2. **Directory handling**
3. **Random access file handling**

(6.1) File handling

1. Tạo mới một project NetBeans

- Chọn **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, gõ tên project **FileInfo**.
- Với trường Create Main Class, gõ **FileInfo**.
- Click **Finish**.

- Project **FileInfo** và file **FileInfo.java** xuất hiện trong IDE NetBeans.

2. Sửa file **FileInfo.java** như đoạn Code-6.11. Chú ý nhưng dòng code in đậm.

```
import java.io.File;
import java.io.IOException;

public class FileInfo {

   public static void main(String[] args) {

      // The first command line argument needs to be provided
      String fileName = args[0];
      File fn = new File(fileName);
      try {
         fn.createNewFile();
      } catch (IOException e) {

      }
```

```
        System.out.println("Name: " + fn.getName());

        // Check if the file exists using exists() method
        if (fn.exists()) {
            System.out.println(fileName + " does exist.");
        }

        if (fn.canRead()) {
            System.out.println(fileName  + " is readable.");
        }

        System.out.println(fileName + " is " + fn.length() + " bytes long.");
        System.out.println(fileName + " is last modifed at " +
              new java.util.Date(fn.lastModified())) ;

        if (fn.canWrite()) {
            System.out.println(fileName  + " is writable.");
        }
        else{
            System.out.println(fileName  + " is not writable.");
        }
    }
}
```

Code-6.11: FileInfo.java

3. Tạo đối số để truyền theo dòng lệnh.

- Click chuột phải lên project **FileInfo** chọn **Properties**.
- Cửa sổ **Project Properties** xuất hiện.
- Chọn Run, trong trường Arguments, gõ **dummyname**.

4. Dịch và chạy project

- Click Chuột phải vào project **FileInfo** và chọn Run.
- Xem kết quả xuất hiện trong cửa sổ Output (Figure-6.12).

```
Name: dummyname
dummyname does exist.
dummyname is readable.
dummyname is 0 bytes long.
dummyname is last modifed at Sat Feb 24 15:36:28 EST 2007
dummyname is writable.
```

Figure-6.12: Result of running UnFileInfo application

return to top of the exercise

(6.2) Directory handling

1. Tạo mới một project NetBeans

- Tại pane Choose Project, chon Java trong Categories và Java Application trong Projects. Click Next.
- Tại pane Name and Location, với trường Project Name, gõ tên project **DirectoryInfo**.
- Với trường Create Main Class, gõ **DirectoryInfo**.
- Click **Finish**.

- Project **DirectoryInfo** và file **DirectoryInfo.java** xuất hiện trong IDE NetBeans.

2. Sửa file **DirectoryInfo.java** như đoạn Code-6.21. Chú ý nhưng dòng code in đậm.

```
import java.io.File;

public class DirectoryInfo {

    public static void main(String[] args) {

        // Create a directory
        System.out.println("Creating temp directory...");
        String fileName = "temp";
        File fn = new File(fileName);
        fn.mkdir();

        // Create sub directories under the temp directory
        File subdir1 = new File(fn, "subdir1");
        subdir1.mkdir();
        File subdir2 = new File(fn, "subdir2");
        subdir2.mkdir();

        // Check if it is a file or directory using isFile() method
        System.out.println(fileName + " is a " +
              (fn.isFile()? "file." :"directory."));

        if (fn.isDirectory()) {
            String content[] = fn.list();
            System.out.println("The content of this directory:");
            for (int i = 0; i < content.length; i++) {
                System.out.println(content[i]);
            }
        }

        // Delete a directory
        System.out.println(fileName +
              (fn.exists()? " exists": " does not exist"));
        System.out.println("Deleting temp directory...");
        fn.delete();

    }
}
```

Code-6.21: DirectoryInfo.java

3 Build and run the project

- Right click **DirectoryInfo** project and select **Run**.
- Observe the result in the **Output** window. (Figure-6.22)

```
Creating temp directory...
temp is a directory.
The content of this directory:
```

```
subdir1
subdir2
temp exists
Deleting temp directory...
```
Figure-6.22: Result

return to top of the exercise

(6.3) Random access file handling

1. Tạo mới một project NetBeans

- Tại pane Choose Project, chon Java trong Categories và Java Application trong Projects. Click Next.
- Tại pane Name and Location, với trường Project Name, gõ tên project **RandomAccessFileHandling**
- Với trường Create Main Class, gõ **RandomAccessFileHandling**.
- Click **Finish**.

- Project **RandomAccessFileHandling** và file **RandomAccessFileHandling.java** xuất hiện trong IDE NetBeans.

2. Sửa file **RandomAccessFileHandling.java** như đoạn Code-6.31. Chú ý nhưng dòng code in đậm.

```
import java.io.IOException;
import java.io.RandomAccessFile;

public class RandomAccessFileHandling {

    public static void main(String[] args) {
        try {
            RandomAccessFile raf = new RandomAccessFile("test.txt", "rw");
            raf.writeInt(10);
            raf.writeInt(43);
            raf.writeInt(88);
            raf.writeInt(455);

            // change the 3rd integer from 88 to 99
            raf.seek((3 - 1) * 4);
            raf.writeInt(99);
            raf.seek(0); // go to the first integer
            int i = raf.readInt();
            while (i != -1) {
                System.out.println(i);
                i = raf.readInt();
            }
            raf.close();
        } catch (IOException e) {
        }
    }
}
```
Code-6.31: RandomAccessFileHandling.java

3 Dịch và chạy project

- Click Chuột phải vào project **RandomAccessFileHandling** và chọn Run.
- Xem kết quả xuất hiện trong cửa sổ Output (Figure-6.32).

```
10
43
99
455
```
Figure-6.32: Result

return to top of the exercise

Summary

Trong bài lab này, bạn đã làm quen với lớp File để thực hiên một số thao tác liên quan đến file và thư mục.

return to the top

Exercise 7: Filter Streams

Trong bài lab này chúng ta sẽ làm quen với các lớp FilterInputStream và FilterOutputStream.

(7.1) Create custom stream class that extends Filter streams

1. Tạo mới một project NetBeans

- Chọn **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, gõ tên project **FilterInputOutputStream**
- Với trường Create Main Class, gõ **FilterInputOutputStream**.
- Click **Finish**.

- Project **FilterInputOutputStream** và file **FilterInputOutputStream.java** xuất hiện trong IDE NetBeans.

2. Sửa file **FilterInputOutputStream.java** như đoạn Code-7.11. Chú ý nhưng dòng code in đậm.

```
import java.io.*;

public class FilterInputOutputStream {

    public static void main(String[] args) throws IOException {

        Adler32 inChecker = new Adler32();
        Adler32 outChecker = new Adler32();
        CheckedInputStream in = null;
        CheckedOutputStream out = null;

        try {
            in = new CheckedInputStream(
```

```
            new FileInputStream("farrago.txt"),
            inChecker);
        out = new CheckedOutputStream(
            new FileOutputStream("outagain.txt"),
            outChecker);
    } catch (FileNotFoundException e) {
        System.err.println("CheckedIOTest: " + e);
        System.exit(-1);
    } catch (IOException e) {
        System.err.println("CheckedIOTest: " + e);
        System.exit(-1);
    }

    int c;

    while ((c = in.read()) != -1)
      out.write(c);

    System.out.println("Input stream check sum: " +
          inChecker.getValue());
    System.out.println("Output stream check sum: " +
          outChecker.getValue());

    in.close();
    out.close();
  }
}
```

Code-7.11: FilterInputOutputStream.java

3. Tạo file CheckedInputStream.java. (Code-7.12 below)

```
import java.io.FilterInputStream;
import java.io.InputStream;
import java.io.IOException;

// Custom InputStream
public class CheckedInputStream extends FilterInputStream {

  private Checksum cksum;

  public CheckedInputStream(InputStream in, Checksum cksum) {
    super(in);
    this.cksum = cksum;
  }

  public int read() throws IOException {
    int b = in.read();
    if (b != -1) {
      cksum.update(b);
    }
    return b;
  }

  public int read(byte[] b) throws IOException {
    int len;
    len = in.read(b, 0, b.length);
    if (len != -1) {
      cksum.update(b, 0, len);
    }
    return len;
  }

  public int read(byte[] b, int off, int len) throws IOException {
    len = in.read(b, off, len);
    if (len != -1) {
      cksum.update(b, off, len);
    }
    return len;
  }

  public Checksum getChecksum() {
    return cksum;
  }
}
```

Code-7.12: CheckedInputStream.java

4. Tạo CheckedOutputStream.java. (Code-7.13 below)

```
import java.io.*;

// Custom OutputStream
public class CheckedOutputStream extends FilterOutputStream {

  private Checksum cksum;

  public CheckedOutputStream(OutputStream out, Checksum cksum) {
    super(out);
    this.cksum = cksum;
  }

  public void write(int b) throws IOException {
    out.write(b);
    cksum.update(b);
  }

  public void write(byte[] b) throws IOException {
    out.write(b, 0, b.length);
    cksum.update(b, 0, b.length);
  }

  public void write(byte[] b, int off, int len) throws IOException {
    out.write(b, off, len);
    cksum.update(b, off, len);
  }

  public Checksum getChecksum() {
    return cksum;
  }
}
```

Code-7.13: CheckedOutputStream.java

5. Tạo file **Checksum.java**. (Figure-8.14 below)

```
public interface Checksum {
    /**
     * Updates the current checksum with the specified byte.
     */
    public void update(int b);

    /**
     * Updates the current checksum with the specified array of bytes.
     */
    public void update(byte[] b, int off, int len);

    /**
     * Returns the current checksum value.
     */
    public long getValue();

    /**
     * Resets the checksum to its initial value.
     */
    public void reset();
}
```

Code-7.14: Checksum.java

6. Tạo file **Adler32.java**. (Code-7.15 below)

```
public class Adler32 implements Checksum {

    private int value = 1;

    /*
     * BASE is the largest prime number smaller than 65536
     * NMAX is the largest n such that 255n(n+1)/2 + (n+1)(BASE-1) <= 2^32-1
     */
    private static final int BASE = 65521;
    private static final int NMAX = 5552;

    /**
     * Update current Adler-32 checksum given the specified byte.
     */
    public void update(int b) {
        int s1 = value & 0xffff;
        int s2 = (value >> 16) & 0xffff;
        s1 += b & 0xff;
        s2 += s1;
        value = ((s2 % BASE) << 16) | (s1 % BASE);
    }

    /**
     * Update current Adler-32 checksum given the specified byte array.
     */
    public void update(byte[] b, int off, int len) {
        int s1 = value & 0xffff;
        int s2 = (value >> 16) & 0xffff;

        while (len > 0) {
            int k = len < NMAX ? len : NMAX;
            len -= k;
            while (k-- > 0) {
                s1 += b[off++] & 0xff;
                s2 += s1;
            }
            s1 %= BASE;
            s2 %= BASE;
        }
        value = (s2 << 16) | s1;
    }

    /**
     * Reset Adler-32 checksum to initial value.
     */
    public void reset() {
        value = 1;
    }

    /**
     * Returns current checksum value.
     */
    public long getValue() {
        return (long)value & 0xffffffff;
    }
}
```

Code-7.15: Checksum.java

7. Tạo file input **farrago.txt**.

```
So she went into the garden to cut a cabbage-leaf, to
make an apple-pie; and at the same time a great
she-bear, coming up the street, pops its head into the
shop. 'What! no soap?' So he died, and she very
imprudently married the barber; and there were
present the Picninnies, and the Joblillies, and the
Garyalies, and the grand Panjandrum himself, with the
little round button at top, and they all fell to playing
the game of catch as catch can, till the gun powder ran
out at the heels of their boots.

Samuel Foote 1720-1777
```

File-2.12: farrago.txt

8. Dịch và chạy project

- Click Chuột phải vào project **FilterInputOutputStream** và chọn Run.
- Xem kết quả xuất hiện trong cửa sổ Output. (Figure-7.16 below)

```
Input stream check sum: 736868089
Output stream check sum: 736868089
```
Figure-7.16: Result

return to top of the exercise

Summary

Trong bài lab này chúng ta sẽ làm quen với các lớp FilterInputStream và FilterOutputStream.

return to the top

Exercise 8: Scanner and Formatter

Trong phần này chúng ta sẽ luyện tập với lớp Scanner và Formatter

1. **Scanner**
2. **Formatter**

(8.1) Scanner class

1. Tạo mới một project NetBeans

- Chọn 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, gõ tên project **ScannerClass**
- Với trường Create Main Class, gõ **ScannerClass**..
- Click Finish.

- Project **ScannerClass** và file **ScannerClass**.java xuất hiện trong IDE NetBeans.

2. Sửa file **ScannerClass**.java như đoạn Code-8.11. Chú ý nhưng dòng code in đậm.

```
import java.io.*;
import java.util.*;

public class ScannerClass {

   public static void main(String[] args) throws IOException {

      Scanner s =
            new Scanner(new BufferedReader(new FileReader("usnumbers.txt")));
      s.useLocale(Locale.US);

      double sum = 0;

      while (s.hasNext()) {
         sum += s.nextDouble();
      }
      s.close();

       System.out.println("Sum of all numbers = " + sum);
   }
}
```
Code-8.11: ScannerClass.java

3. Tạo file input **usnumbers.txt**.

```
8.5
32,767
3.14159
1,000,000.1
```
File-9.12: usnumbers.txt

4. Dịch và chạy project

- Click Chuột phải vào project **ScannerClass** và chọn Run.
- Xem kết quả xuất hiện trong cửa sổ Output (Figure-8.12).

```
Sum of all numbers = 1032778.74159
```
Figure-8.12: Result

return to top of the exercise

(8.2) Format class

1. Tạo mới một project NetBeans

- Chọn 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, gõ tên project **FormatClass**
- Với trường Create Main Class, gõ **FormatClass**.
- Click Finish.

- Project **FormatClass** và file **FormatClass.java** xuất hiện trong IDE NetBeans.

2. Sửa file **FormatClass.java** như đoạn Code-8.21. Chú ý nhưng dòng code in đậm.

```
public class FormatClass {
   public static void main(String[] args) {
      System.out.format("%f, %1$+020.10f %n", Math.PI);
   }
}
```
Code-8.21: FormatClass.java

3. Dịch và chạy project

- Click Chuột phải vào project **FormatClass** và chọn Run.
- Xem kết quả xuất hiện trong cửa sổ Output (Figure-8.22).

```
3.141593, +00000003.1415926536
```
Figure-8.22: Result

return to top of the exercise

Summary

Trong phần này chúng ta đã luyện tập với lớp Scanner và Formatter

return to the top

Luyện tập

1. **Viết chương trình kiểm tra nội dung 2 file text có giống nhau hay không? Tên file nhập vào từ bàn phím.**
2. **Viết chương trình tạo file text out.txt nếu file chưa tồn tại, sau đó sinh 100 số nguyên ngẫu nhiên và ghi xuống file, các số nguyên cách nhau bởi một khoảng trống.**
3. **Viết chương trình thêm 100 số nguyên ngẫu nhiên vào một file text đã tồn tại, các số nguyên cách nhau bởi một ký tự trống.**
4. **Viết chương trình tạo một file nhị phân out.dat và ghi một lượng ngẫu nhiên các số nguyên ngẫu nhiên xuống file theo kiểu nhị phân.**
5. **Viết chương trình đọc một file nhị phân chứa các số nguyên (số lượng số nguyên không biết trước) cách nhau bởi ký tự trống, và in ra tổng các số.**