การพัฒนาเว็บเซอร์วิส JAX-WS ที่มีชนิดข้อมูลแบบซับซ้อนด้วย NetBeans

นายชัยวัฒน์ บุตรไชย

ชนิดข้อมูลในเว็บเซอร์วิสแบ่งออกเป็น 2 ชนิดใหญ่ๆ ได้แก่ ชนิดข้อมูลแบบพื้นฐานที่อ้างอิงตามโครงสร้างเอกสาร XML (primitive datatypes and derived datatypes) และชนิดข้อมูลแบบซับซ้อน (complex datatypes) ซึ่งในการพัฒนาเว็บเซอร์วิสที่มีชนิดข้อมูลพื้นฐานเพียงอย่างเดียวอาจไม่เพียงพอต่อการใช้งานเว็บเซอร์วิสได้ จำเป็นต้องอาศัยชนิดข้อมูลแบบซับซ้อนในการอธิบายข้อมูลด้วย

ชนิดข้อมูลแบบพื้นฐาน (Primitive datatypes and Derived datatypes) เป็นชนิดข้อมูลพื้นฐานที่ได้ถูกอธิบายโดยองค์กร W3C ซึ่งได้กลายเป็นมาตรฐานของ XML Schema (http://www.w3.org/TR/2000/CR-xmlschema-2-20001024/) และชนิดข้อมูลพื้นฐานส่วนใหญ่จะถูกจับคู่กับชนิดข้อมูลพื้นฐานของโปรแกรมต่างๆ เช่น boolean, byte, short, int, long, float, double, string เป็นต้น ซึ่งข้อมูลเหล่านี้สามารถถูกใช้เป็น ชื่อซึ่งมีชนิดข้อมูลเป็นข้อความ (string), อายุซึ่งมีชนิดข้อมูลแบบตัวเลข (int), ที่อยู่ซึ่งมีชนิดข้อมูลแบบข้อความ (string)

```
<name>string</name>
<age>int</age>
<address>string</address>
```

ชนิดข้อมูลแบบซับซ้อน (Complex datatypes) เป็นชนิดข้อมูลที่ผู้ใช้งานสร้างขึ้นมาเอง โดยมีพื้นฐานหรือหน่วยชนิดข้อมูลที่ย่อยที่สุดเป็นชนิดข้อมูลแบบพื้นฐาน เช่น ชนิดข้อมูลของสมุดโทรศัพท์ (เป็นชนิดข้อมูลแบบซับซ้อน) ประกอบด้วย ชื่อมีชนิดข้อมูลแบบข้อความ (string), ที่อยู่มีชนิดข้อมูลแบบข้อความ (string) และเบอร์โทรศัพท์มีชนิดข้อมูลแบบข้อความ (string) ทั้งสามข้อมูลเป็นชนิดข้อมูลแบบพื้นฐานที่อยู่ภายใต้ชนิดข้อมูลแบบซับซ้อน

```
<contact>
      <name>string</name>
      <address>string</address>
      <phone>string</phone>
</contact>
```

ชนิดข้อมูลพื้นฐานในภาษาจาวาเทียบได้กับชนิดข้อมูลแบบพื้นฐานใน XML Scheman รวมถึง String, Date, Time เป็นต้น และถ้าจะกล่าวถึงชนิดข้อมูลแบบซับซ้อน “POJO” หรือ “Plain Old Java Object” ถือว่าเทียบได้กับชนิดข้อมูลแบบซับซ้อน ดังนั้น ในเอกสารนี้จะกล่าวถึงการใช้ NetBeans ในการออกแบบและสร้าง POJO เพื่อเป็นชนิดข้อมูลให้กับเว็บเซอร์วิส

เอกสารนี้นำเสนอถึงการสร้างเว็บเซอร์วิส JAX-WS ให้มีชนิดข้อมูลแบบซับซ้อนด้วย NetBeans ซึ่งสามารถที่จะรับข้อมูลที่เป็นชนิดพื้นฐานหรือข้อมูลชนิดซับซ้อน และส่งข้อมูลกลับเป็นชนิดพื้นฐานหรือข้อมูลชนิดซับซ้อนได้

ความต้องการพื้นฐานสำหรับเอกสาร

- Java Standard Development Kit (JDK) version 5.0 (http://java.sun.com)
- NetBeans IDE 5.5.1 + Enterprise Pack 5.5 (http://www.netbeans.org)
- Sun Java System Application Server (included in Enterprise Pack 5.5)

ขั้นตอนหลักในการพัฒนาเว็บเซอร์วิสให้มีชนิดข้อมูลแบบซับซ้อน

1. ออกแบบชนิดข้อมูลที่ต้องการ
2. พัฒนา POJO ตามข้อมูลที่ได้ออกแบบไว้
3. พัฒนาเว็บเซอร์วิส และใช้ชนิดข้อมูลของ POJO
4. ประกาศเว็บเซอร์วิส

ออกแบบชนิดข้อมูลที่ต้องการ

เอกสารนี้จะยกตัวอย่างข้อมูลของ contacts ที่ประกอบไปด้วย contact หลายๆ ตัว โดยแต่ละตัวมี name, address และ phone ซึ่ง phone ยังประกอบไปด้วย mobile, office, home และ other เมื่อเขียนเป็นเอกสาร XML สามารถเขียนได้ดังนี้

```
<contacts>
	<contact>
		<name>string</name>
		<address>string</address>
		<phone>
			<mobile>string</mobile>
			<home>string</home>
		</phone>
	</contact>
		.
		.
		.
	<contact>
		<name>string</name>
		<address>string</address>
		<phone>
			<mobile>string</mobile>
			<home>string</home>
		</phone>
	</contact>
</contacts>
```

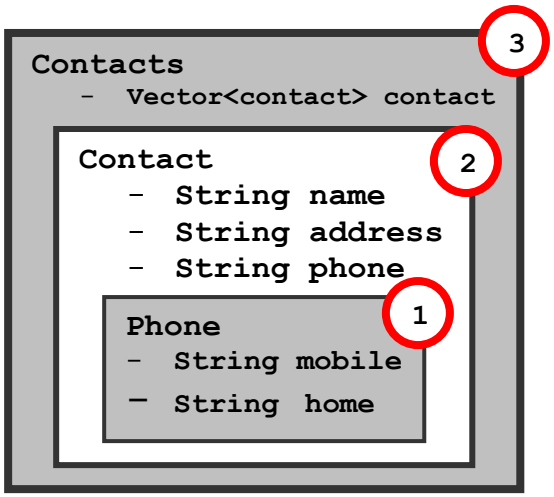
หรือถ้าจะเทียบกับ XML Schema ก็จะเป็นดังนี้

```
<xsd:element name="contacts" type="tns:contactsType"/>
<xsd:element name="contact" type="tns:contactType"/>
<xsd:complexType name="contactsType">
    <xsd:sequence>
        <xsd:element name="contact" type="tns:contactType"
                maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="contactType">
    <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="address" type="xsd:string"/>
        <xsd:element name="phone" type="tns:phoneType"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="phoneType">
    <xsd:sequence>
        <xsd:element name="mobile" type="xsd:string"/>
        <xsd:element name="home" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
```

พัฒนา POJO ตามข้อมูลที่ได้ออกแบบไว้

การสร้าง POJO คือการสร้างออปเจ็คจาวา เมื่อเทียบกับ XML ที่ได้ออกแบบไว้ จะต้องสร้างออปเจ็คจาวา จากชนิดข้อมูลพื้นฐานเสียก่อน จึงสร้างชนิดข้อมูลแบบซับซ้อนเรียกใช้ชนิดข้อมูลพื้นฐานอีกที

(ก)



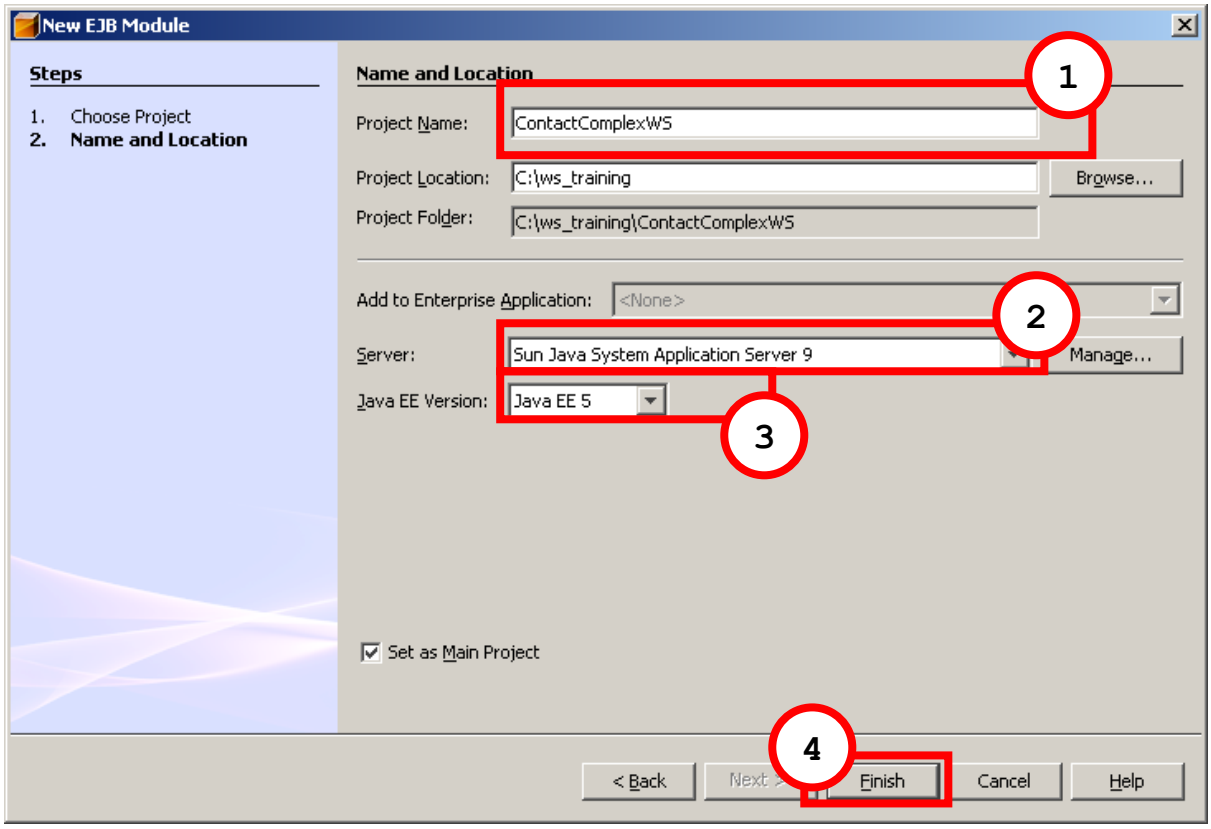
(ข)

```
<xsd:element name="contacts" type="tns:contactsType"/>
<xsd:element name="contact" type="tns:contactType"/>
<xsd:complexType name="contactsType">
    <xsd:sequence>
        <xsd:element name="contact" type="tns:contactType"
            maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="contactType">
    <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="address" type="xsd:string"/>
        <xsd:element name="phone" type="tns:phoneType"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="phoneType">
    <xsd:sequence>
        <xsd:element name="mobile" type="xsd:string"/>
        <xsd:element name="home" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
```

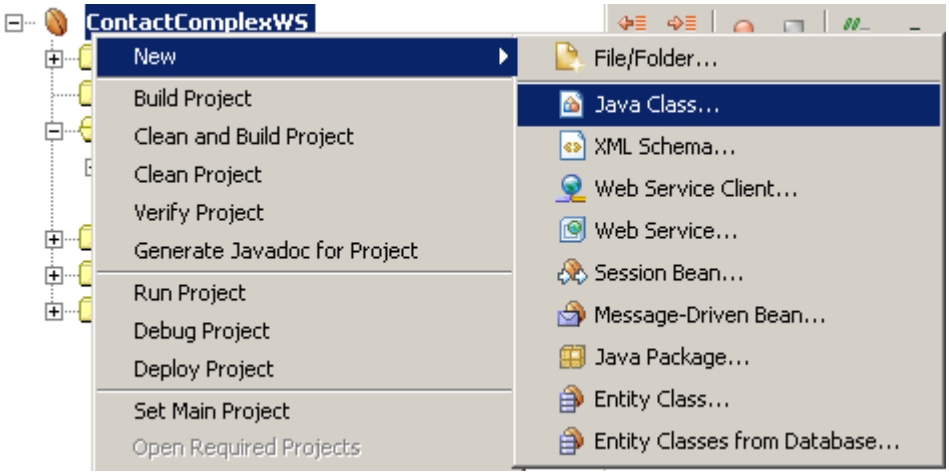
**(ก) แต่ละกล่องหมายถึงแต่ละ public class เหตุผลที่ทำคลาสซ้อนกัน เพราะต้องการแสดงให้เห็นถึงการเทียบได้ข้อมูลที่ได้ออกแบบไว้ และตัวเลขในวงกลม หมายถึงลำดับการสร้างคลาสของจาวา*

1. สร้างโปรเจคที่เป็น EJB Module (เตรียมสร้าง JAX-WS) File > New Project
2. ในฝั่ง Categories เลือกเป็น Enterprise และฝั่ง Projects เลือกเป็น EJB Module จากนั้นคลิก Next

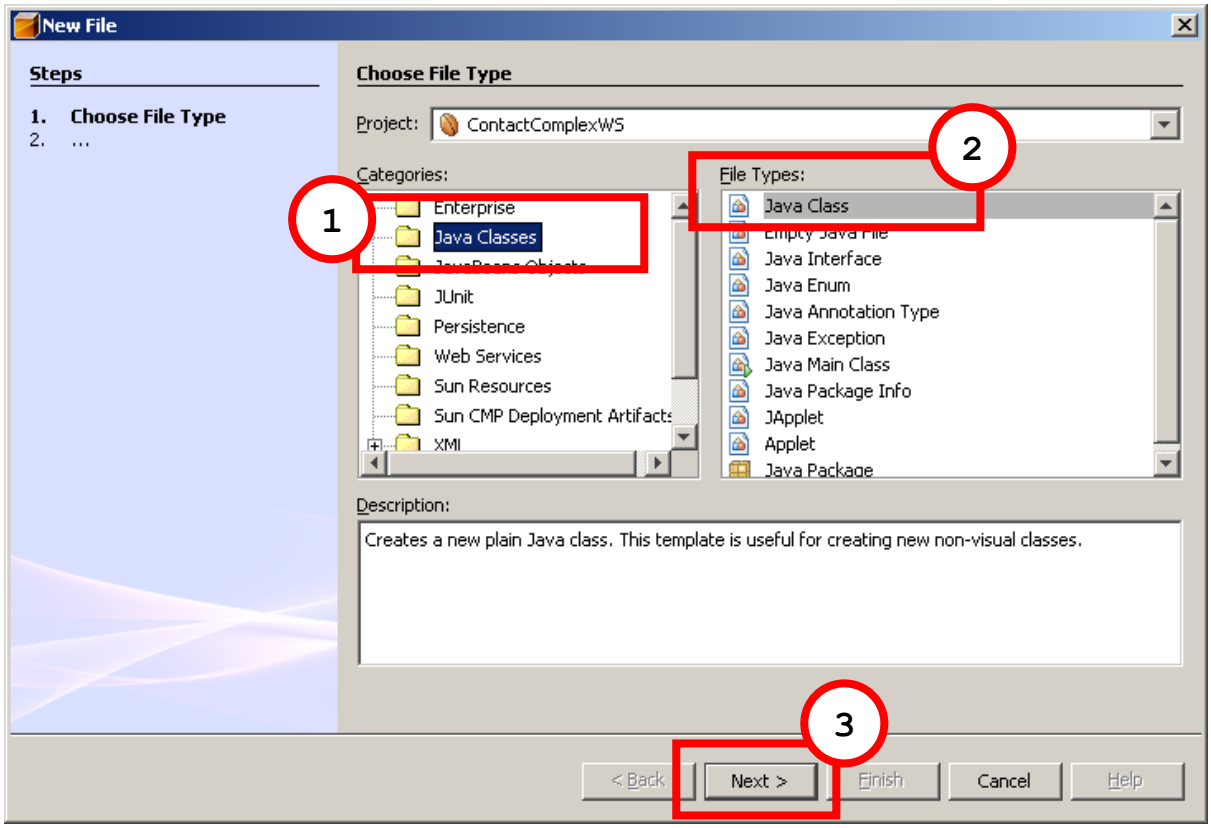
3. ตั้งชื่อโปรเจคเป็น ContactComplexWS โดยใช้ Server เป็น Sun Java System Application Server 9 และเลือก Java EE Version เป็น Java EE 5



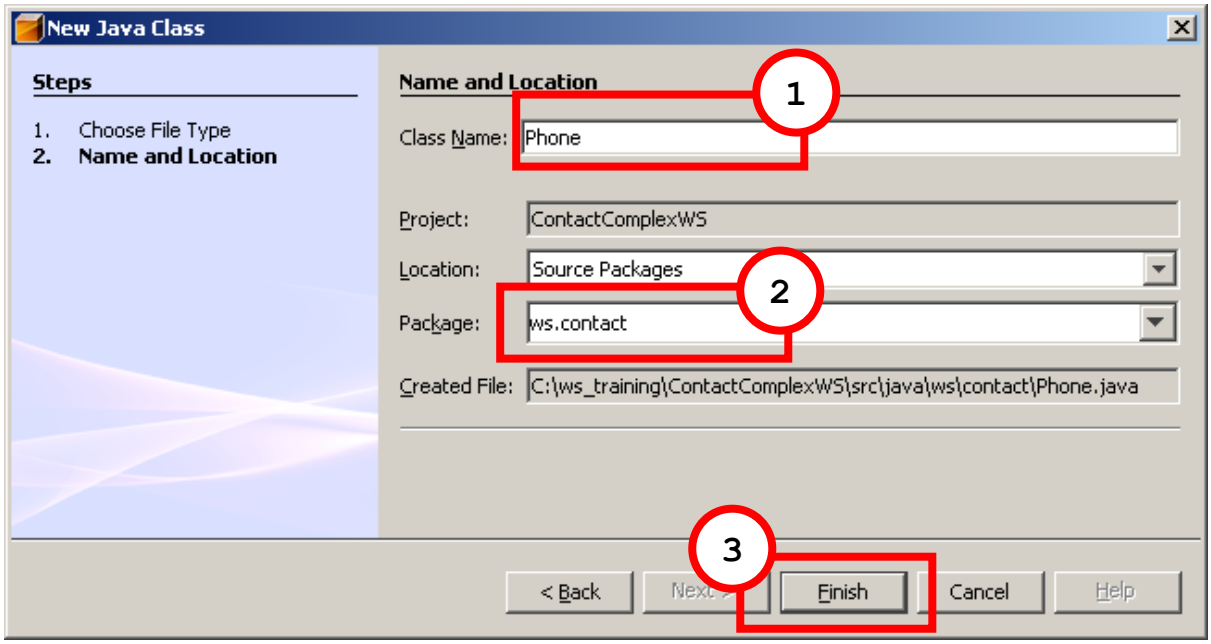
4. สร้างคลาสจาวา ด้วยการคลิกขวาที่โปรเจค แล้วเลือก Java Class



หากไม่พบ ให้เลือกเป็น File/Folder… แล้วเลือกฝั่ง Catagories เป็น Java Classes และฝั่ง File Types เป็น Java Class



5. ตั้งชื่อคลาสใน Class Name ชื่อ Phone และใส่ชื่อ Package เป็น ws.contact



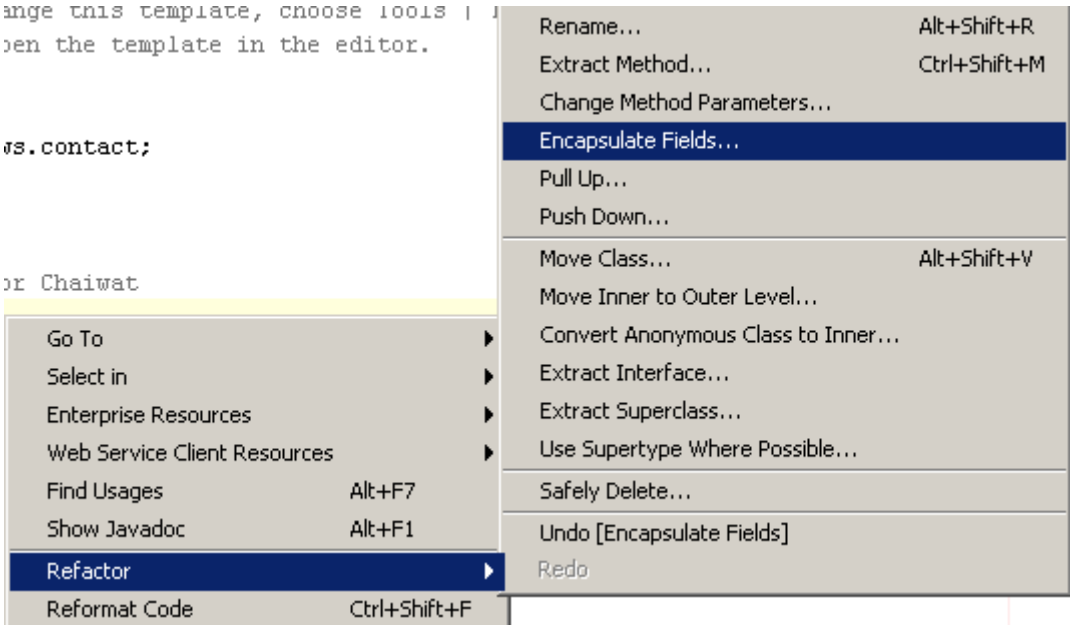
6. เพิ่มแอตทริบิวต์ mobile, home ที่มีชนิดข้อมูลเป็น String ลงใน Phone.java ดังนี้

```
1               public class Phone {
2
3                   /** Creates a new instance of Phone */
4                   public Phone() {
5                   }
6
7               }
```

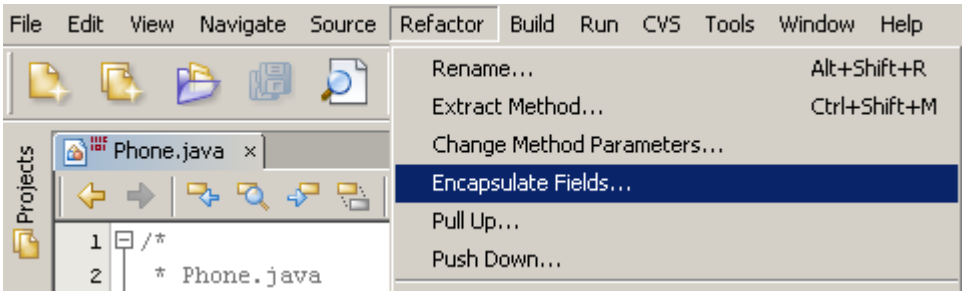
เป็น

```
1               public class Phone {
2
3                   /** Creates a new instance of Phone */
4                   public Phone() {
5                   }
6
7                   private String mobile;
8                   private String home;
9
10              }
```

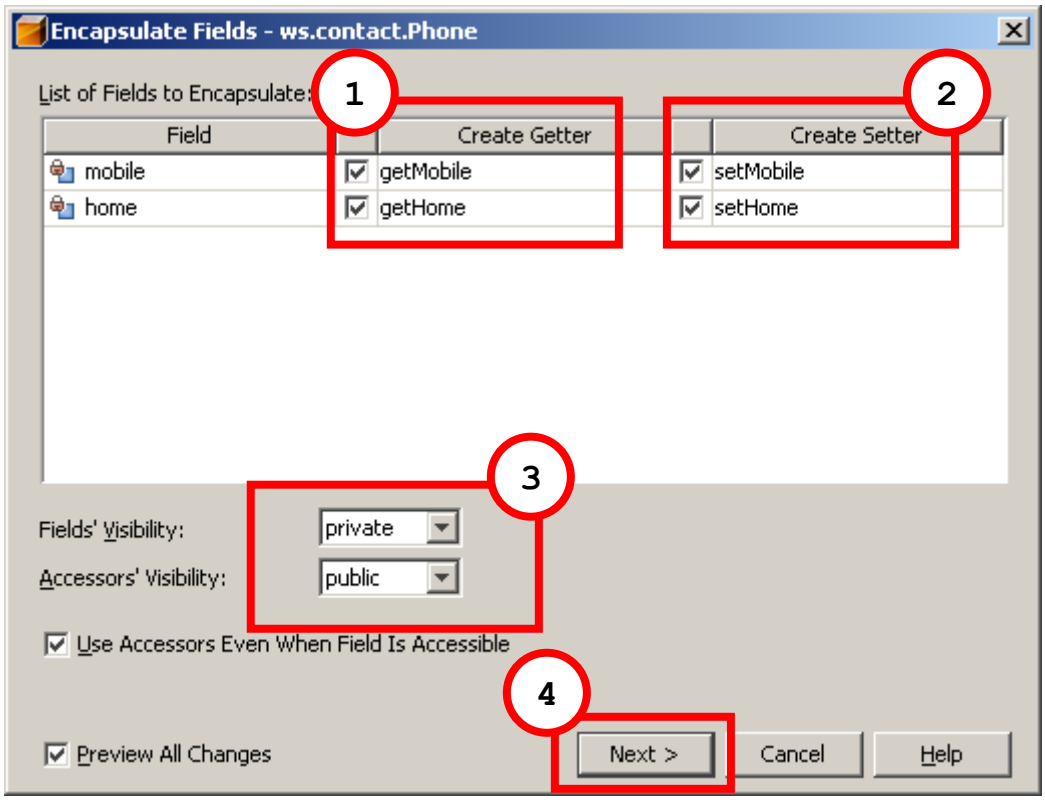
7. สร้างเมธอดแบบ getter และ setter ให้กับแอตทริบิวต์ที่สร้างขึ้น โดยคลิกขวาที่บรรทัดที่ 9 ในข้อ 6 แล้วเลือก Refactor > Encapsulate Fields



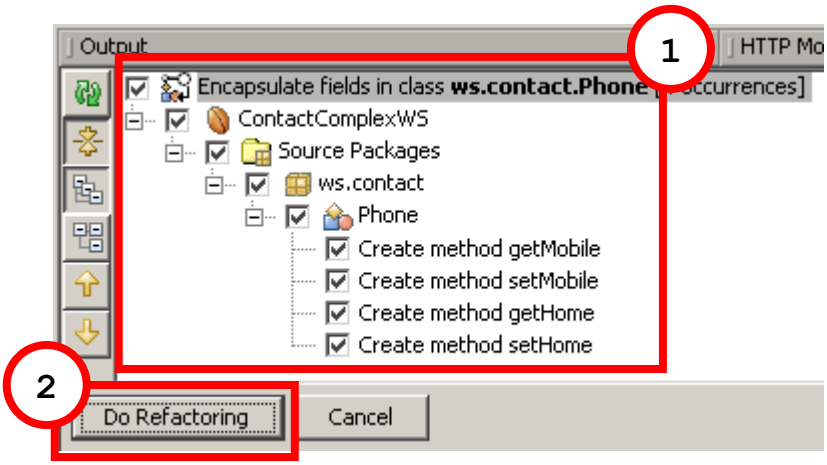
หรือคลิก Refactor ที่เมนู แล้วเลือก Encapsulate Field



8. ในหน้าต่าง Encapsulate Fields ให้ตรวจสอบว่าโปรแกรมจะสร้างทั้ง Getter และ Setter



9. ที่ด้านล่างของ NetBeans (หน้าจอ Output) จะมีการถามยืนยันและตรวจสอบเพื่อให้แน่ใจอีกครั้งในการ Refactoring



10. NetBeans จะทำการสร้างเมธอดตามที่ได้ระบุไว้ในข้อ 8 และข้อ 9 ซึ่งคลาส Phone.java ใช้เพียงเท่านี้
11. สร้างคลาสจาวาใหม่ตามลำดับในรูป (ก) และตั้งชื่อว่า Contact ไว้ในแพ็กเกจ ws.contact

12. เพิ่มแอตทริบิวต์ name, address และ phone ที่มีชนิดข้อมูลเป็น String, String และ Phone ตามลำดับ ใน Contact.java ดังนี้

```
1               public class Contact {
2
3                   /** Creates a new instance of Contact */
4                   public Contact() {
5                   }
6
7               }
```

เป็น

```
1               public class Contact {
2
3                   /** Creates a new instance of Contact */
4                   public Contact() {
5                   }
6
7                   private String name;
8                   private String address;
9                   private Phone phone;
10
11              }
```

13. ทำตามขั้นตอนที่ 7 - 9 อีกครั้งสำหรับคลาส Contact.java เพื่อให้เสร็จสมบูรณ์
14. สร้างคลาสจาวาใหม่ตามลำดับในรูป (ก) และตั้งชื่อว่า Contacts ไว้ในแพ็กเกจ ws.contact
15. เพิ่มแอตทริบิวต์ contact ที่มีชนิดข้อมูลเป็น Vector<Contact> ใน Contacts.java ดังนี้

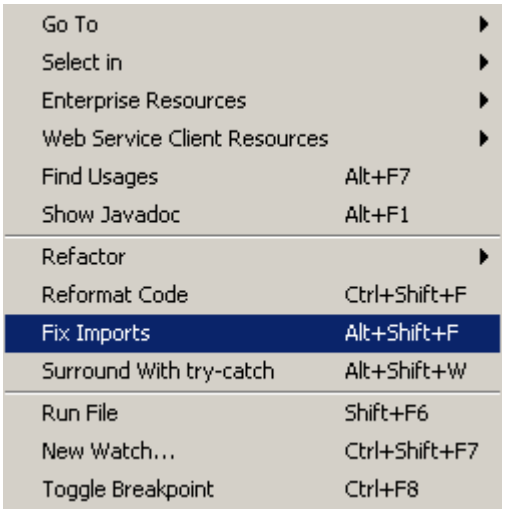
```
1               public class Contacts {
2
3                   /** Creates a new instance of Contacts */
4                   public Contacts() {
5                   }
6
7               }
```

เป็น

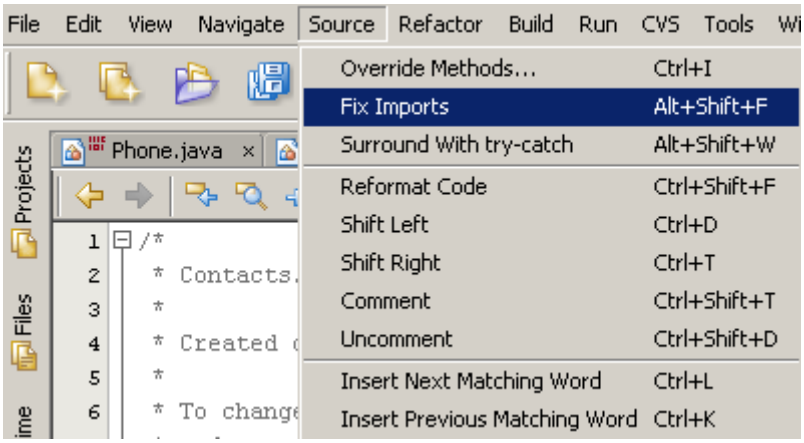
```
1               public class Contacts {
2
3                   /** Creates a new instance of Contacts */
4                   public Contacts() {
5                   }
6
7                   private Vector<Contact> contact;
8
9               }
```

**เนื่องจากได้ออกแบบให้ Contacts สามารถมี Contact ได้มากกว่า 1 Contact ดังนั้นแอตทริบิวต์ที่จะเพิ่มควรเป็นชนิดข้อมูลประเภท array หรือ collection แต่ในที่นี้แนะนำเป็น vector (ประเภท collection) เนื่องจากมีการใช้งานที่ง่ายกว่า อีกทั้งการแปลงข้อมูลไปเป็น WSDL สำหรับ vector ให้ผลไม่แตกต่างจาก array*

16. เพิ่มการ import java.util.Vector; สำหรับชนิดข้อมูลของ Vector ด้วยการคลิกขวาที่พื้นที่ว่างใน Contacts.java แล้วเลือกที่ Fix Imports



หรือสามารถคลิกที่ Source บนเมนู แล้วเลือกที่ Fix Imports



17. เพิ่มเมธอด addContact() ที่มี return type เป็น Contact สำหรับข้อมูล vector เพื่อให้สามารถเพิ่มข้อมูลได้ง่าย ดังนี้

```
1       public void setContact(Vector<Contact> contact) {
2           this.contact = contact;
3       }
4
5   }
```

เป็น

```
1       public void setContact(Vector<Contact> contact) {
2           this.contact = contact;
3       }
4
5       public Contact addContact() {
6           Contact c = new Contact();
7           if (contact == null) {
8               contact = new Vector<Contact>();
9           }
10          contact.add(c);
11          return contact.lastElement();
12      }
13
14      public void addContact(Contact c) {
15          if (contact == null) {
16              contact = new Vector<Contact>();
17          }
18          contact.add(c);
19      }
20
21  }
```

บรรทัดที่ 6 สร้าง Temporary ตัวแปรสำหรับการเพิ่มเข้าไปใน Contact

บรรทัดที่ 7 – 9 ตรวจสอบว่า ค่าเริ่มต้นของ Contact มีหรือไม่ ถ้าไม่มีให้สร้างไว้ (ค่าจะได้ไม่เป็น null)

บรรทัดที่ 10 นำค่าในบรรทัดที่ 6 ที่เป็นตัวแปร Temp มาใส่ไว้

บรรทัดที่ 11 ส่งค่าตัวแปร Temp ไปให้ตัวแปรที่ร้องขอ จากนั้นตัวแปรที่ร้องขอสามารถเพิ่มค่าต่างๆ ได้อย่างง่ายดาย

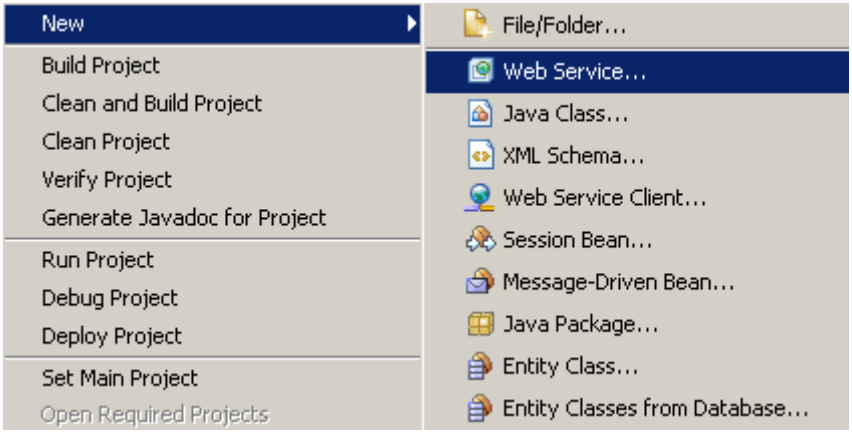
พัฒนาเว็บเซอร์วิส และใช้ชนิดข้อมูลของ POJO

การสร้าง Operation ของ JAX-WS จะเหมือนกับการสร้างเมธอดในภาษาจาวา ซึ่งเมื่อสร้างเมธอดเสร็จแล้ว JAX-WS จะแปลงเมธอดเหล่านั้นให้เป็น WSDL เพื่อเป็นเอกสารอธิบายตัวเมธอดของ JAX-WS เอง และในส่วนของข้อมูลภายในเมื่อมีการรับค่า หรือส่งค่าใดๆ JAX-WS จะทำหน้าที่จัดการแปลงให้อยู่ในรูปของ XML เองทั้งหมด ดังนั้นการสร้างเว็บเซอร์วิสด้วยวิธีนี้จึงเหมือนกับการสร้างเมธอดทั่วๆ ไป

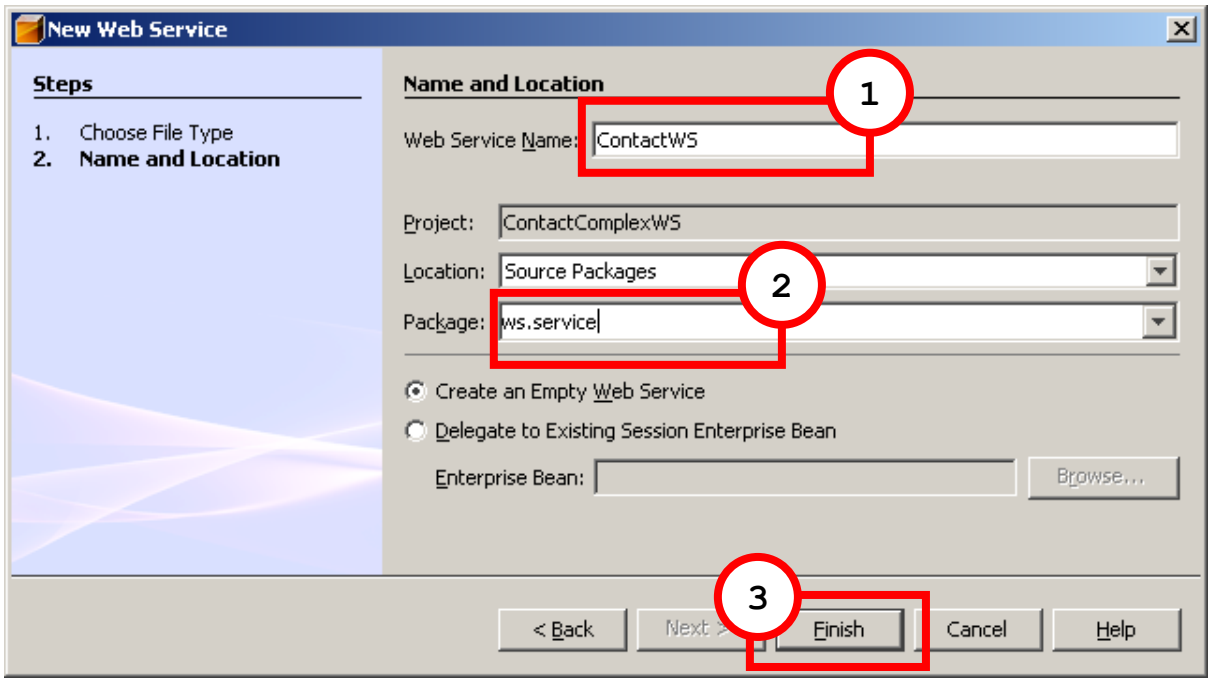
ในตัวอย่างจะนำเสนอถึงการสร้าง Operation ต่างๆ ที่มีการรับ และส่งแตกต่างกัน ดังนี้

Operation Name	Request datatypes (input)	Response datatypes (output)
GetAllContacts	-	Contacts
AddContact	Contact	String
GetContactByName	String	Contact

1. สร้างเว็บเซอร์วิสในโปรเจค โดยคลิกขวา New > Web Service



2. ตั้งชื่อเว็บเซอร์วิสเป็น ContactWS และให้อยู่ในแพ็คเกจ ws.service



กำหนดค่าเริ่มต้นให้กับ Contact

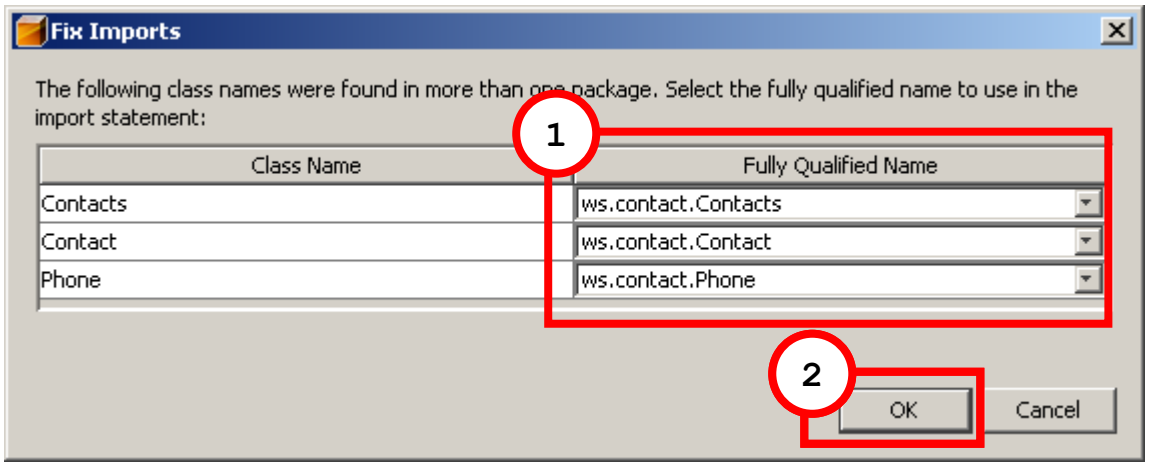
3. เพิ่มโค้ดด้านล่างนี้ลงในโปรแกรม

```
1  public class ContactWS {
2
3
4
5  }
```

เป็น

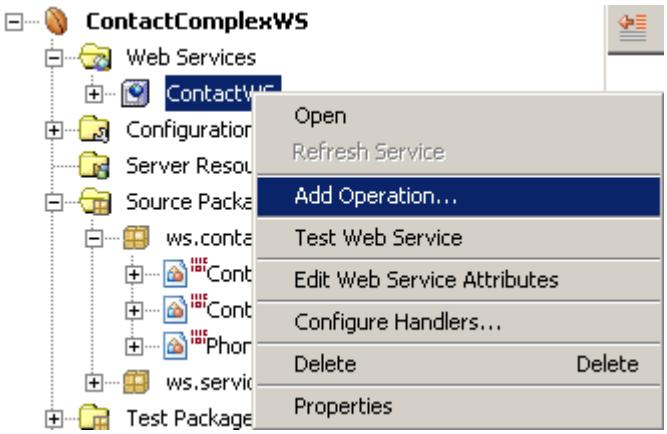
```
1  public class ContactWS {
2
3      private static Contacts contacts = new Contacts();
4      static {
5          initContacts();
6      }
7
8      private static Contacts initContacts() {
9          // Add Somchai to Contact
10         Contact somchai = contacts.addContact();
11         somchai.setName("Somchai");
12         somchai.setAddress("BKK");
13         Phone ps = new Phone();
14         ps.setHome("025512222");
15         ps.setMobile("0815552222");
16
17         somchai.setPhone(ps);
18
19         // Add A to Contact
20         Contact chaiwat = contacts.addContact();
21         chaiwat.setName("Chaiwat");
22         chaiwat.setAddress("KKN");
23         Phone pc = new Phone();
24         pc.setHome("043222888");
25
26         chaiwat.setPhone(pc);
27
28         // Both Somchai and Chaiwat are now added in Contacts
29         return contacts;
30     }
31
32 }
```

4. ทำการ Fix Imports เพื่อแก้ไขข้อผิดพลาดที่อาจเกิดขึ้น โดยการคลิกขวา > Fix Imports

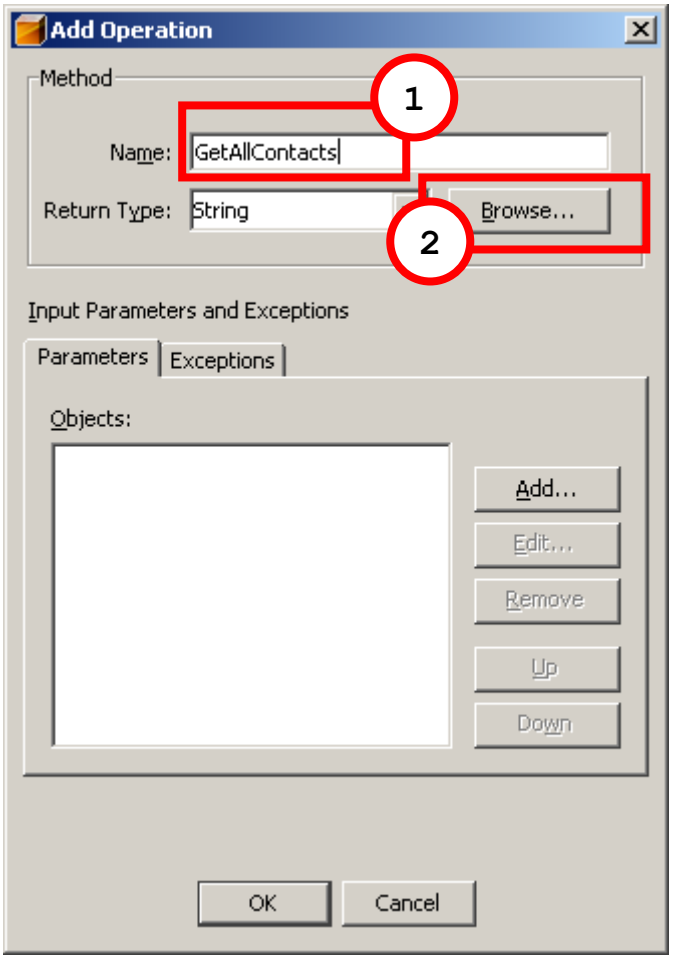


สร้าง Operation ชื่อ GetAllContacts

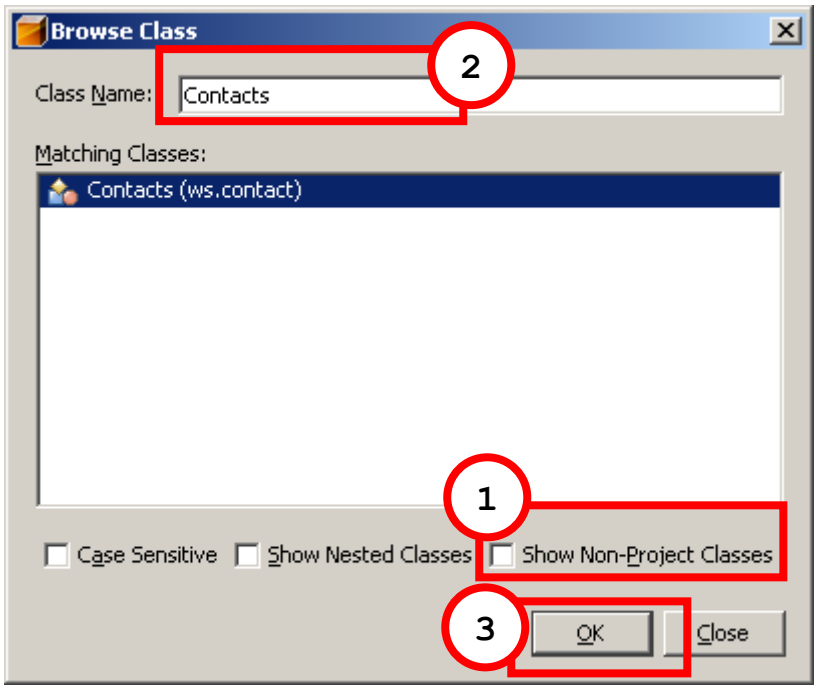
5. เพิ่ม Operation ให้เว็บเซอร์วิส โดยคลิกขวาที่ ContactWS ใต้โหนด Web Services ในโปรเจค ContactComplexWS



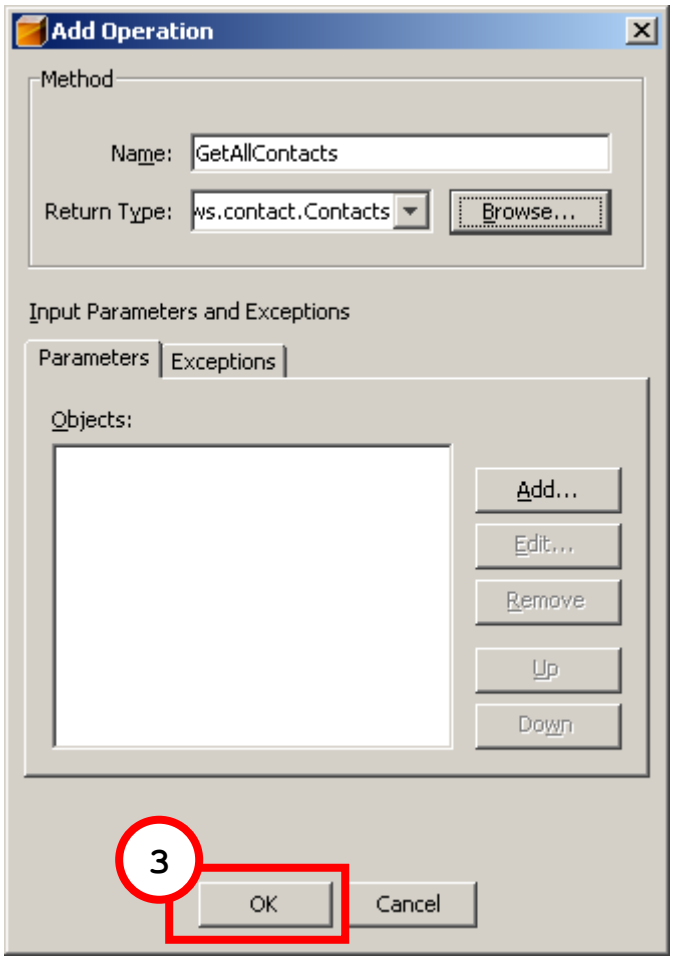
6. ในหน้าต่าง Add Operation ในส่วนของ Method ให้ใช้ Name เป็น GetAllContacts และที่ Return Type ให้คลิก Browse… เพื่อเลือกออปเจ็คจาวา



7. หน้าต่าง Browse Class ให้คลิกเครื่องหมายถูกออกจากช่อง Show Non-Project Classes และในช่อง Class Name ให้ใส่เป็น Contacts



8. ในส่วนของ Input Parameters and Exception ไม่ต้องใส่



9. แก้ไขโค้ดที่ถูกสร้างขึ้นมาให้ส่งค่ากลับเป็น contacts ดังนี้

```
    /**
     * Web service operation
     */
    @WebMethod
    public Contacts GetAllContacts() {
        // TODO implement operation
        return null;
    }

}
```

เป็น

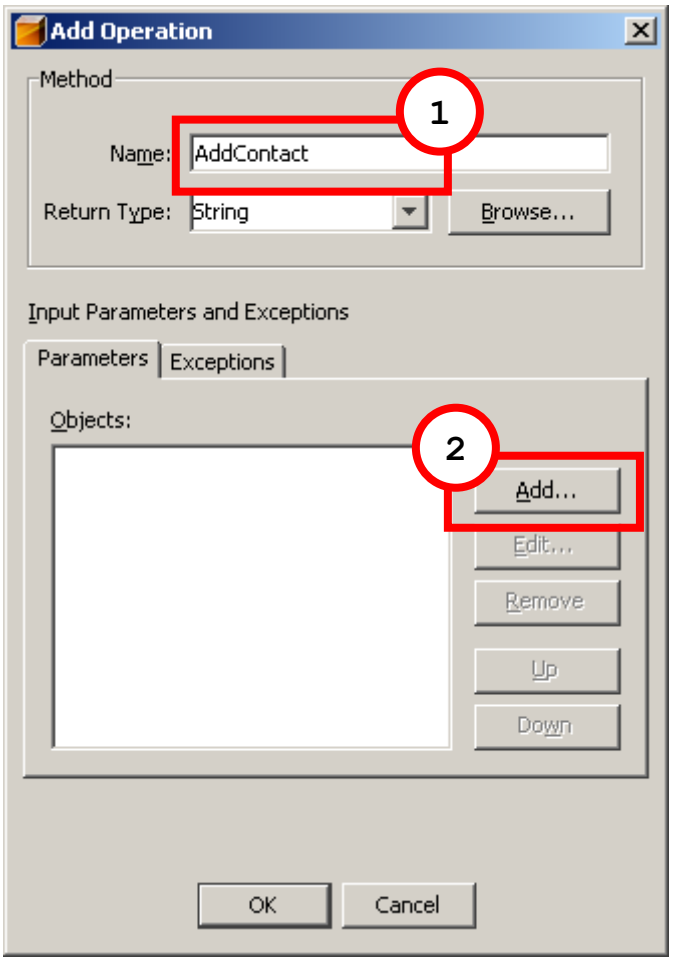
```
    /**
     * Web service operation
     */
    @WebMethod
    public Contacts GetAllContacts() {
        // TODO implement operation
        return contacts;
    }

}
```

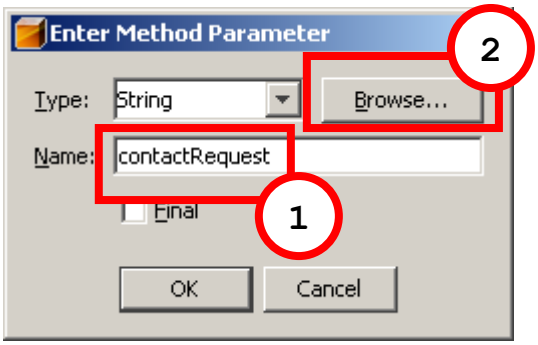
**เมื่อสิ้นสุดข้อ 9 และไม่มีข้อผิดพลาดใดๆ แจ้งขึ้น อาจจะทดลองประกาศเว็บเซอร์วิสและทดสอบเว็บเซอร์วิสดูได้*

สร้าง Operation ชื่อ AddContact

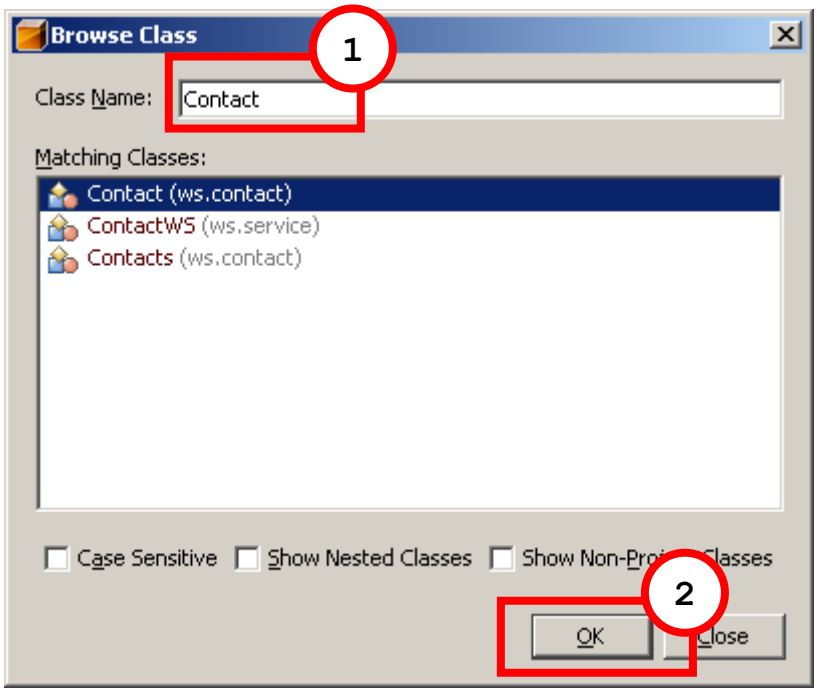
10. ทำตามข้อ 5 และ 7 โดยตั้งชื่อเมธอดว่า AddContact แล้วทำการเพิ่ม Input Parameters and Exceptions



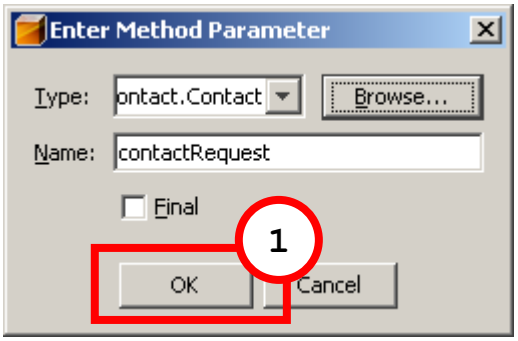
11. ใส่ชื่อ Name เป็น contactRequest จากนั้นทำการ Browser… หา Type (คล้ายกับข้อ 6)



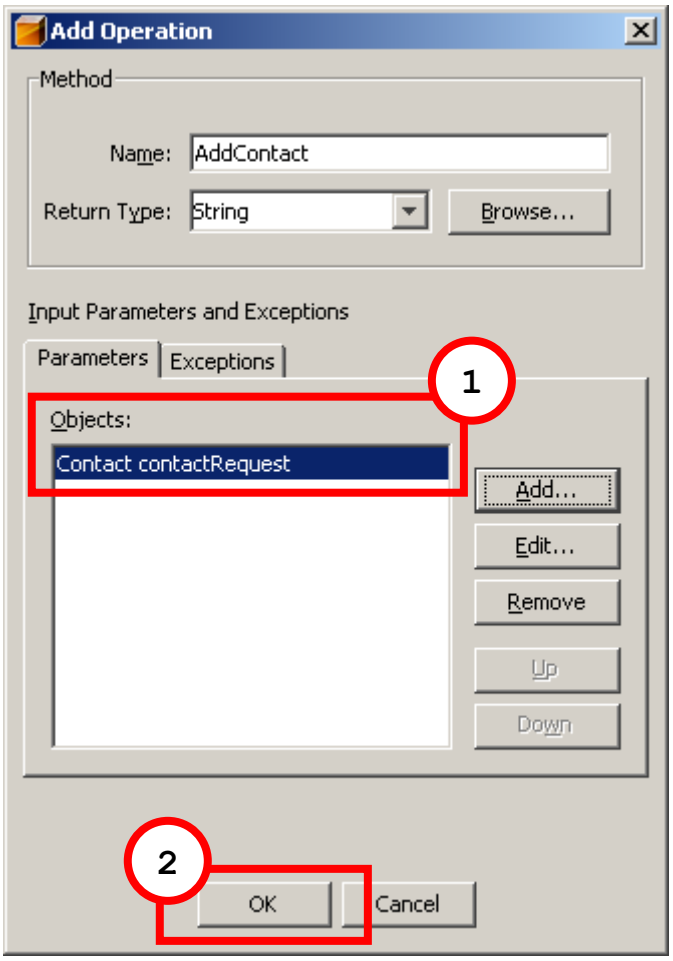
12. ในหน้าต่าง Browse Class ใส่ชื่อ Contact ลงใน Class Name



13. คลิก OK เมื่อได้ Type ที่ต้องการ



14. ส่วนของ Input Parameters and Exceptions จะปรากฏออปเจคที่เพิ่มขึ้นมา



15. แก้ไขโค้ดของ AddContact ดังนี้

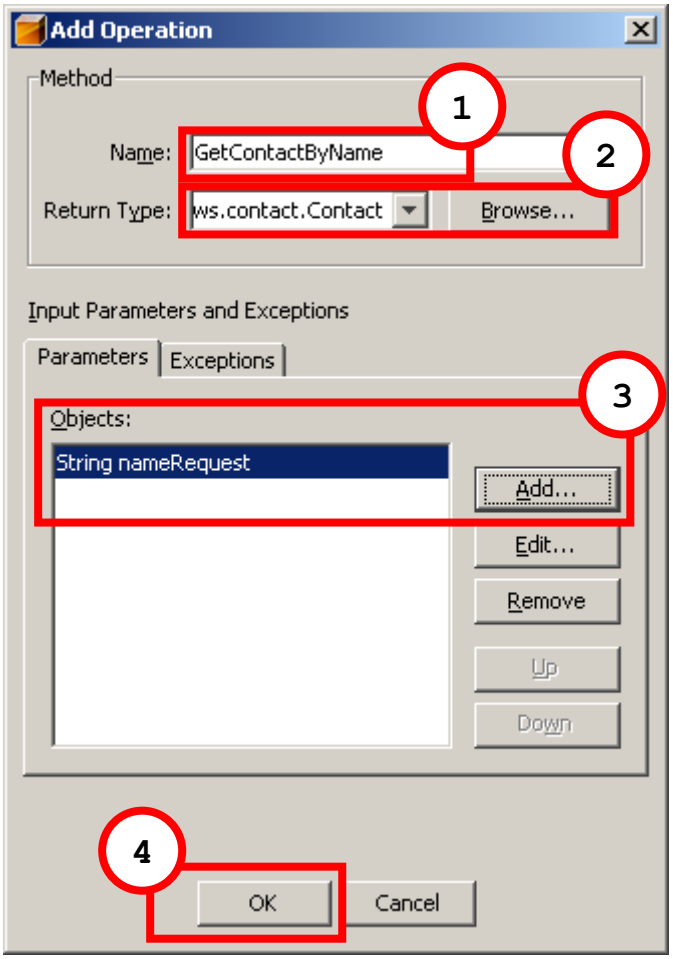
```
    public String AddContact(@WebParam(name = "contactRequest") Contact
contactRequest) {
        // TODO implement operation
        return null;
    }
```

เป็น

```
    public String AddContact(@WebParam(name = "contactRequest") Contact
contactRequest) {
        // TODO implement operation
        if (contactRequest == null ||
                contactRequest.getName() == null) {
            return "The contact can't be added.";
        } else {
            contacts.addContact(contactRequest);
            return "The contact has been added.";
        }
    }
```

สร้าง **Operation ชื่อ GetContactByName**

16. ทำตามข้อ 5 – 7 และ 10 – 14 โดยตั้งชื่อเมธอดเป็น GetContactByName โดยมี Return Type เป็น Contact และมี Input Parameters เป็น String มีชื่อว่า nameRequest



17. แก้ไขโค้ดของ GetContactByName ดังนี้

```
1       public Contact GetContactByName(@WebParam(name = "nameRequest") String
    nameRequest) {
2           // TODO implement operation
3           return null;
4       }
```

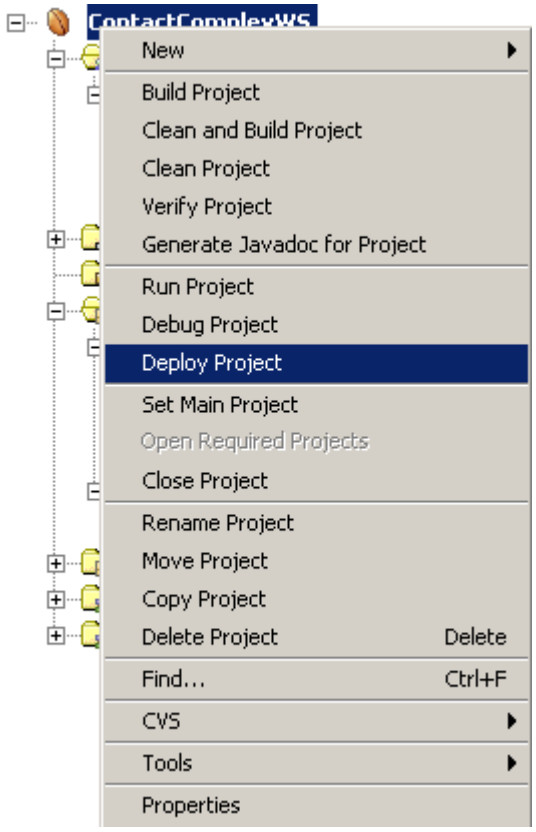
เป็น

```
1       public Contact GetContactByName(@WebParam(name = "nameRequest") String
    nameRequest) {
2           // TODO implement operation
3           Vector<Contact> contact = contacts.getContact();
4           for (Contact c : contact)
5               if (c.getName().equals(nameRequest))
6                   return c;
7           return new Contact();
8       }
```

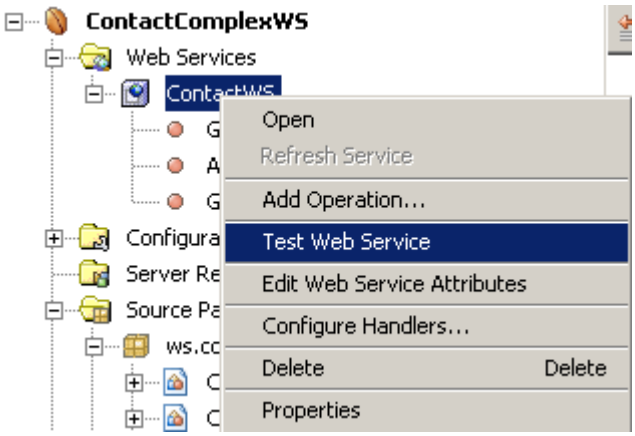
18. แก้ไขข้อผิดพลาดด้วยการ Fix Imports (คลิกขวา > Fix Imports)

ประกาศเว็บเซอร์วิส

1. คลิกขวาที่ Project > Deploy Project



2. ทดสอบเว็บเซอร์วิสอย่างง่าย โดยคลิกขวาที่ ContactWS ใต้โหนด Web Services ในโปรเจค ContactComplexWS แล้วเลือกที่ Test Web Service



3. NetBeans จะทำการเปิดบราวส์เซอร์ที่เป็นการเรียกใช้อย่างง่ายขึ้นมา ซึ่งจะสามารถเรียกใช้ข้อมูลที่ร้องขอเป็นชนิดข้อมูลแบบพื้นฐานได้เท่านั้น (ดังนั้นจึงไม่สามารถทดสอบ Operation AddContact ได้)


```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://service.ws/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    targetNamespace="http://service.ws/" name="ContactWSService">
  - <types>
    - <xsd:schema>
        <xsd:import namespace="http://service.ws/"
          schemaLocation="http://move2atom:8080/ContactWSService/ContactWS/__container$publishing$subctx/META-
          INF/wsdl/ContactWSService_schema1.xsd" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
          xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" />
      </xsd:schema>
    </types>
    <part name="parameters" element="tns:GetAllContacts" />
  </message>
  - <message name="GetAllContactsResponse">
    <part name="parameters" element="tns:GetAllContactsResponse" />
  </message>
  - <message name="AddContact">
    <part name="parameters" element="tns:AddContact" />
  </message>
  - <message name="AddContactResponse">
    <part name="parameters" element="tns:AddContactResponse" />
  </message>
  - <message name="GetContactByName">
    <part name="parameters" element="tns:GetContactByName" />
  </message>
  - <message name="GetContactByNameResponse">
    <part name="parameters" element="tns:GetContactByNameResponse" />
  </message>
  - <portType name="ContactWS">
    - <operation name="GetAllContacts">
        <input message="tns:GetAllContacts" />
        <output message="tns:GetAllContactsResponse" />
      </operation>
    - <operation name="AddContact">
        <input message="tns:AddContact" />
        <output message="tns:AddContactResponse" />
      </operation>
    - <operation name="GetContactByName">
        <input message="tns:GetContactByName" />
        <output message="tns:GetContactByNameResponse" />
      </operation>
```

Import XML Schema ของ POJO

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <xs:schema version="1.0" targetNamespace="http://service.ws/" xmlns:tns="http://service.ws/"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="AddContact" type="tns:AddContact" />
    <xs:element name="AddContactResponse" type="tns:AddContactResponse" />
    <xs:element name="GetAllContacts" type="tns:GetAllContacts" />
    <xs:element name="GetAllContactsResponse" type="tns:GetAllContactsResponse" />
    <xs:element name="GetContactByName" type="tns:GetContactByName" />
    <xs:element name="GetContactByNameResponse" type="tns:GetContactByNameResponse" />
  - <xs:complexType name="GetContactByName">
    - <xs:sequence>
        <xs:element name="nameRequest" type="xs:string" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  - <xs:complexType name="GetContactByNameResponse">
    - <xs:sequence>
        <xs:element name="return" type="tns:contact" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  - <xs:complexType name="contact">
    - <xs:sequence>
        <xs:element name="address" type="xs:string" minOccurs="0" />
        <xs:element name="name" type="xs:string" minOccurs="0" />
        <xs:element name="phone" type="tns:phone" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  - <xs:complexType name="phone">
    - <xs:sequence>
        <xs:element name="home" type="xs:string" minOccurs="0" />
        <xs:element name="mobile" type="xs:string" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="GetAllContacts" />
  - <xs:complexType name="GetAllContactsResponse">
    - <xs:sequence>
        <xs:element name="return" type="tns:contacts" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  - <xs:complexType name="contacts">
    - <xs:sequence>
        <xs:element name="contact" type="tns:contact" nillable="true" maxOccurs="unbounded" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  - <xs:complexType name="AddContact">
    - <xs:sequence>
        <xs:element name="contactRequest" type="tns:contact" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  - <xs:complexType name="AddContactResponse">
    - <xs:sequence>
        <xs:element name="return" type="xs:string" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  </xs:schema>
```

XML Schema ของ Contact POJO

2 getAllContacts Method invocation

Method parameter(s)

Type	Value

Method returned

ws.service.Contacts : "**ws.service.Contacts@e301e0**"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmls
<soapenv:Body>
<ns1:GetAllContacts/>
</soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmls
<soapenv:Body>
<ns1:GetAllContactsResponse>
<return>
<contact>
<address>BKK</address>
<name>Somchai</name>
<phone>
<home>025512222</home>
<mobile>0815552222</mobile>
</phone>
</contact>
<contact>
<address>KKN</address>
<name>Chaiwat</name>
<phone>
<home>043222888</home>
</phone>
</contact>
</return>
</ns1:GetAllContactsResponse>
</soapenv:Body>
</soapenv:Envelope>
```

3 getContactByName Method invocation

Method parameter(s)

Type	Value
java.lang.String	Chaiwat

Method returned

ws.service.Contact : "**ws.service.Contact@a5bc81**"

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
<soapenv:Body>
<ns1:GetContactByName>
<nameRequest>Chaiwat</nameRequest>
</ns1:GetContactByName>
</soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/
<soapenv:Body>
<ns1:GetContactByNameResponse>
<return>
<address>KKN</address>
<name>Chaiwat</name>
<phone>
<home>043222888</home>
</phone>
</return>
</ns1:GetContactByNameResponse>
</soapenv:Body>
</soapenv:Envelope>
```

**ทุก Operation อาจจะทดสอบโดยใช้ soapUI ที่ WSDL URL: http://localhost:8080/ContactWSService/ContactWS?WSDL*