Advanced Java Programming Lab Sheet
III Year /VI Part
Faculty: BCA

Lab sheet 4

Objectives:

1. To execute and create JavaBean.
2. To familiarize with JavaBean Bound and constrained properties.
3. To understand concept of JavaBean persistence.

Objective 1:

```
package com.texas.javabeansdemo;
import java.io.Serializable;

public class Student implements Serializable {
	private static final long serialVersionUID = 1L;
	private int ID;
	private String firstName;
	private String middleName;
	private String lastName;
	private long[] mobileNo;
	private boolean handicap;

	public Student() {

	}
	public int getID() {
		return ID;
	}
	public void setID(int iD) {
		ID = iD;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getMiddleName() {
		return middleName;
	}
	public void setMiddleName(String middleName) {
		this.middleName = middleName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public long[] getMobileNo() {
		return mobileNo;
	}
	public void setMobileNo(long[] mobileNo) {
		this.mobileNo = mobileNo;
	}
	public boolean isHandicap() {
		return handicap;
	}
	public void setHandicap(boolean handicap) {
		this.handicap = handicap;
	}
}
```

Assignment:

1.0. Write a list of JavaBean properties used in above code.
1.1. Execute the above program and set the value of all properties and display the result using the get method.
1.2. Write a JavaBean program that finds square and cube value in terms of properties.

Objective 2:

Source bean:

```
package com.texas.boundproperty;
import java.io.Serializable;
import java.beans.*;

public class Parent implements Serializable {

    private static final long serialVersionUID = 1L;
    private String firstName = "Shankar";
    private String lastName = "Dahal";
    private PropertyChangeSupport pcs;

    public Parent() {
            pcs = new PropertyChangeSupport(this);
    }
    public void setFirstName(String firstName) {
            this.firstName = firstName;
    }
    public String getFirstName() {
            return firstName;
    }
    public void setLastName(String newLastName) {
            String oldLastName = lastName;
            pcs.firePropertyChange("LastName", oldLastName, newLastName);
            this.lastName = newLastName;
    }
    public String getLastName() {
            return lastName;
    }
    public void addPropertyChangeListener(PropertyChangeListener pcl) {
            pcs.addPropertyChangeListener(pcl);

    }
    public void removePropertyChangeListener(PropertyChangeListener pcl) {
            pcs.removePropertyChangeListener(pcl);
    }
}
```

The following bean is a listener that wants to be notified when the lastName property of the Parent bean changes. So, it implements the PropertyChangeListener interface and defines the method propertyChange(). In the propertyChange() method, it changes its own lastName property to the new value of the lastName property of the Parent bean by using the getNewValue() method on the PropertyChangeEvent object.

Child Bean :

```
package com.texas.boundproperty;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.io.Serializable;

public class Child implements Serializable, PropertyChangeListener {

	private static final long serialVersionUID = 1L;
	private String firstName = "Ram";
	private String lastName = "Dahal";

	public Child() {
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getLastName() {
		return lastName;
	}
	public void propertyChange(PropertyChangeEvent evt) {

		if(evt.getPropertyName().equals("LastName")) {
			String newValue = evt.getNewValue().toString();
			System.out.println("New Value in Child Bean.");
			setLastName(newValue);
		}
	}
}
```

For demonstration purposes, we have created the following Java application program.

```
package com.texas.boundproperty;

public class MainBoundDemo {

	public static void main(String[] args) {

		Parent parent = new Parent();
		Child child = new Child();
		parent.addPropertyChangeListener(child);

		System.out.println("Before Change");
		System.out.println("Parent : " + parent.getFirstName() + ", " + parent.getLastName());
		System.out.println("Child : " + child.getFirstName() + ", " + child.getLastName());

		parent.setLastName("pd. Dahal");
		System.out.println("After Change");

		System.out.println("Parent : " + parent.getFirstName() + ", " + parent.getLastName());
		System.out.println("Child : " + child.getFirstName() + ", " + child.getLastName());
	}
}
```

Assignment:

2.0. Write a JavaBean program and modify the district name using JavaBean Bound property.

Constrained:

Source Bean:

```
package com.texas.constrained;
import java.io.Serializable;
import java.beans.*;

public class Broker implements Serializable {
	private static final long serialVersionUID = 1L;
	private int price = 1200;
	private VetoableChangeSupport vcs;

	public Broker() {
		vcs = new VetoableChangeSupport(this);
	}
	public void setPrice(int newPrice) throws PropertyVetoException {
		int oldPrice = price;
		this.price = newPrice;
		vcs.fireVetoableChange("Price", oldPrice, newPrice);
	}
	public int getPrice() {
		return price;
	}
	public void addVetoableChangeListener(VetoableChangeListener vcl) {
		vcs.addVetoableChangeListener(vcl);
	}
	public void removeVetoableChangeListener(VetoableChangeListener vcl) {
		vcs.removeVetoableChangeListener(vcl);
	}
}
```

The ShareHolder is the listener bean. It decides whether the Broker can sell shares at the specified price. It allows the Broker selling shares if the proposed price is greater than or equal to to 1210. Otherwise, it disallows, by throwing the PropertyVetoException object.

```
package com.texas.constrained;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyVetoException;
import java.beans.VetoableChangeListener;

public class ShareHolder implements VetoableChangeListener {

	public void vetoableChange(PropertyChangeEvent evt) throws PropertyVetoException {
		if (evt.getPropertyName().equals("Price")) {
			int newPrice = (int) evt.getNewValue();
			if (!(newPrice >= 1210)) {
				throw new PropertyVetoException("Not interested.", evt);
			}
		}
	}
}
```

For demonstration purposes, we have created the following Java application program.

```
package com.texas.constrained;
import java.beans.PropertyVetoException;

public class MainConstrainedDemo {
	public static void main(String[] args) {

		Broker broker = new Broker();
		ShareHolder shareHolder = new ShareHolder();
		broker.addVetoableChangeListener(shareHolder);

		System.out.println("Before Change ");
		System.out.println("Old price : " + broker.getPrice());

		try {
			broker.setPrice(1209);
			System.out.println("New price : " + broker.getPrice());
		} catch (PropertyVetoException e) {
			System.out.println("Message : " + e.getMessage());
		}
	}
}
```

Assignment:

2.1. Write a JavaBean program for students of Texas International college, if the student has scored more than 40 in Advanced JavaProgramming then set the student as passed otherwise fail. Use JavaBean Constrained.

Objective 3:

Factorial bean.

```
package com.texas.javabeansdemo;
import java.io.Serializable;

public class Factorial implements Serializable {
	private static final long serialVersionUID = 1L;
	private int n;
	protected int factorial;
	public Factorial() {

	}
	public int getN() {
		return n;
	}
	public void setN(int n) {
		this.n = n;
		int fact = 1;
		for (int i = 1; i <= n; i++) {
			fact = fact * i;
		}
		factorial = fact;
	}
	public int getFactorial() {
		return factorial;
	}
}
```

The following code shows how to save the state of a bean.

```
Factorial factorial = new Factorial();
for (int i = 1; i <= 10; i++) {
        factorial.setN(i);
        System.out.println("Factorial of " + i + " is " + factorial.getFactorial());
        if (i > Math.random() * 10) {
                break;
        }
}

// Save
FileOutputStream fileOutputStream = new FileOutputStream("FactorialBean.dat");
ObjectOutputStream objectOutputStream = new ObjectOutputStream(fileOutputStream);
objectOutputStream.writeObject(factorial);
objectOutputStream.close();
```

The following code shows how to retrieve the state of a bean.

```
FileInputStream fileInputStream = new FileInputStream("FactorialBean.dat");
ObjectInputStream objectInputStream = new ObjectInputStream(fileInputStream);
Factorial factorial2 = (Factorial) objectInputStream.readObject();
int n = factorial2.getN();
System.out.println(n);
objectInputStream.close();
```

Assignment:

3.0. Write a JavaBean program and check if odd or even number and save then retrieve the state of the bean using JavaBean persistence.