JAVA ASSIGNMENT – JAVA BEANS

1. What is a JavaBean?

A JavaBean is a reusable software component that follows specific conventions, used for encapsulating data in Java.

2. What are the key features of a JavaBean?

Key features:

- It must have a public no-argument constructor.
- It should be serializable.
- It must provide getter and setter methods to access its properties.
- It follows the convention of having properties as private fields.

3. How do you create a simple JavaBean?

```
public class Person implements Serializable {
   private String name;

   public Person() {}

   public String getName() { return name; }
   public void setName(String name) { this.name = name; }

}
```

4. What is the significance of the no-argument constructor in a JavaBean?

It allows easy instantiation of the JavaBean and is required for frameworks that create instances dynamically, like Java EE and JavaFX.

5. What are getter and setter methods in a JavaBean?

Getter retrieves the value of a property.
Setter sets the value of a property.

6. Explain the naming convention for properties in Java Beans.

Naming convention for properties:

- Property names are written in camelCase.
- Getter method for a property starts with get (e.g., getName()).
- Setter method for a property starts with set (e.g., setName()).
- Boolean properties use **is** for getters (e.g., isActive()).

7. What is the role of the Serializable interface in JavaBeans?

It enables JavaBeans to be serialized, meaning their state can be saved and restored, allowing them to be persisted or transferred over networks.

8. How do you access JavaBean properties?

You access properties via their getter and setter methods, like bean.getProperty() and bean.setProperty(value).

9. What is the Beaninfo class used for?

The BeanInfo class provides metadata about the JavaBean, including its properties, methods, and events, which can be used by design tools.

10. What is the Introspector class in JavaBeans?

The Introspector class is used to analyze a JavaBean and gather information about its properties, methods, and events, typically used by GUI development tools or frameworks.

11. What is the difference between a JavaBean and a regular Java class?

A JavaBean follows specific conventions like having a no-argument constructor, being serializable, and providing getter/setter methods for its properties, while a regular Java class may not adhere to these conventions.

12. Explain the concept of bound properties in JavaBeans.

Bound properties in JavaBeans notify listeners when their value changes. They use PropertyChangeSupport to register and notify listeners of property changes.

13. What are constrained properties in JavaBeans?

Constrained properties allow validation before the value is changed. They use the VetoableChangeListener to check and possibly reject the new value of a property.

14. How do you implement event handling in JavaBeans?

Event handling is implemented by defining event source methods (e.g., addListener, removeListener) and firing events through custom methods, usually involving PropertyChangeSupport for bound properties or VetoableChangeSupport for constrained properties.

15. What is the PropertyChangeListener interface, and how is it used?

PropertyChangeListener listens for changes in bound properties. It is used to receive notifications when a property value changes by implementing the propertyChange method. Example:

```
public void propertyChange(PropertyChangeEvent evt) {
    // Handle the property change
}
```

16. What is the VetoableChangeListener interface, and how is it used?

The VetoableChangeListener interface is used to listen for changes in constrained properties. It allows listeners to veto (prevent) changes to a property before they are applied, using the vetoableChange method.

17. How do you use JavaBeans in JSP?

JavaBeans can be used in JSP by declaring the bean using the <jsp:useBean> tag and accessing its properties via the <jsp:getProperty> and <jsp:setProperty> tags.

```
<jsp:useBean id="person" class="com.example.Person" />
<jsp:setProperty name="person" property="name" value="John" />
<jsp:getProperty name="person" property="name" />
```

18. What is the PersistenceDelegate class used for in JavaBeans?

The PersistenceDelegate class is used to control how JavaBeans are serialized, especially for complex objects. It defines how a JavaBean should be serialized and deserialized, typically within the JavaBeans Introspector or XMLEncoder classes.

19. How do you customize property editors for JavaBeans?

You can customize property editors by implementing the PropertyEditor interface or extending PropertyEditorSupport. This allows you to control how properties are edited, such as converting between string values and the property type.

```
public class MyPropertyEditor extends PropertyEditorSupport {
    public String getAsText() {
        return ((MyType) getValue()).toString();
    }
    public void setAsText(String text) throws IllegalArgumentException {
        setValue(new MyType(text));
    }
}
```

20. Explain the use of DesignMode in Java Beans.

The DesignMode flag indicates whether the JavaBean is being used in a design environment (like an IDE) or a runtime environment. In design mode, you can customize behavior, such as suppressing event firing or providing design-time metadata. For example, bean.isDesignTime() can check if the bean is in design mode.