Java Enterprise Edition

ANUDIP FOUNDATION

JavaBean

Objective: • JavaBean class • Various Properties	**Materials Required:** 1. Eclipse / IntelliJ/ STC
Theory:20mins	**Practical:30mins**
Total Duration: 60 mins	

JavaBean

A JavaBean is a Java class that should follow the following conventions:

- It should have a no-arg constructor.
- It should be Serializable.
- It should provide methods to set and get the values of the properties, known as getter and setter methods.

Why use JavaBean?

According to Java white paper, it is a reusable software component. A bean encapsulates many objects into one object so that we can access this object from multiple places. Moreover, it provides easy maintenance.

Simple example of JavaBean class

```
//Employee.java

package mypack;
public class Employee implements java.io.Serializable{
private int id;
private String name;
public Employee(){}
public void setId(int id){this.id=id;}
public int getId(){return id;}
public void setName(String name){this.name=name;}
public String getName(){return name;}
}
```

How to access the JavaBean class?

To access the JavaBean class, we should use getter and setter methods.

```
package mypack;
public class Test{
public static void main(String args[]){
Employee e=new Employee();//object is created
e.setName("Arjun");//setting value to the object
System.out.println(e.getName());
}}
```

Note: There are two ways to provide values to the object. One way is by constructor and second is by setter method.

JavaBean Properties

A JavaBean property is a named feature that can be accessed by the user of the object. The feature can be of any Java data type, containing the classes that you define.

A JavaBean property may be read, write, read-only, or write-only. JavaBean features are accessed through two methods in the JavaBean's implementation class:

1. getPropertyName ()

For example, if the property name is firstName, the method name would be getFirstName() to read that property. This method is called the accessor.

2. setPropertyName ()

For example, if the property name is firstName, the method name would be setFirstName() to write that property. This method is called the mutator.

Advantages of JavaBean

The following are the advantages of JavaBean:/p>

- The JavaBean properties and methods can be exposed to another application.

- It provides an easiness to reuse the software components.

Disadvantages of JavaBean

The following are the disadvantages of JavaBean:

- JavaBeans are mutable. So, it can't take advantages of immutable objects.
- Creating the setter and getter method for each property separately may lead to the boilerplate code.

OR

A Java Bean is a java class that should follow the java bean standards and requirements. If a java class meets the java bean specification, the java class is called as java bean. The java bean has some properties and characteristics.

A JavaBean is a java class which should follow the specification/requirements/standards/properties below

- Java class should have a public default constructor / no argument constructor.
- All Java-class member variables should be private and non-static.
- All the member variables should have public getter and setter methods, that are used to set and get values.
- Java class is optionally Serializable.

Explanation

The java bean class must have public default constructor. The default constructor is used to create bean object programmatically. The java program dynamically creates an object and assigns a value using the setter methods.

All the member variables must be private and not static, thus the member variables can not be accessed out side of the class even in the inherited class. The only way to access the member variable is to use the Java Bean Getter and Setter methods.

Java beans are referred to as carrier objects. Java beans carry data through several layers of the applic ation. The java bean must be serialised if it is transmitted as data streams to any network or input outp ut system.

Java Bean Example

Student.java

```
package com.yawintutor;

public class Student {
	private int id;
	private String name;
	private boolean active;

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

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

	public boolean isActive() {
		return active;
	}

	public void setActive(boolean active) {
		this.active = active;
	}
}
```

The above example shows Student Class JavaBean which contains three member variables student id, student name and active. Please notice that boolean variables are accessed by prefixing "is" rather than "get".

The main method below shows how to use setter and getter methods to access the java bean.

StudentMain.java

```
package com.yawintutor;

public class StudentMain {
	public static void main(String[] args) {
		Student student = new Student();

		//setting bean values
		student.setId(101);
		student.setName("Karthik");
		student.setActive(true);

		//getting bean value
		System.out.println("Student Id     : "+student.getId());
		System.out.println("Student name   : "+student.getName());
		System.out.println("Student active : "+student.isActive());
	}
}
```

Output

```
Student Id     : 101
Student name   : Karthik
Student active : true
```

Arrays and JavaBeans
A JavaBean can have arrays and can be accessed using the getter and setter method as shown below. Student java bean class has one or more subjects stored in String Array.

Student.java

```
package com.yawintutor;

public class Student {
	private int id;
	private String name;
	public String[] subjects;

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
```

```
        }

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

        public String[] getSubjects() {
                return subjects;
        }

        public void setSubjects(String[] subjects) {
                this.subjects = subjects;
        }

}
```

StudentMain.java

```
package com.yawintutor;

public class StudentMain {
        public static void main(String[] args) {
                Student student = new Student();
                String[] subjects = { "English", "Science", "Computer" };

                // setting bean values
                student.setId(101);
                student.setName("Karthik");
                student.setSubjects(subjects);

                // getting bean value
                System.out.println("Student Id      : " + student.getId());
                System.out.println("Student name    : " + student.getName());

                String[] subjectArray = student.getSubjects();
                for (int i = 0; i < subjectArray.length; i++) {
                        System.out.println("Student subject " + (i + 1) + " : " + subjectArray[i]);
                }
        }
}
```

Output

```
Student Id      : 101
Student name    : Karthik
Student subject 1 : English
Student subject 2 : Science
```

Student subject 3 : Co JavaBeans with Collections object

A Java Bean can have Java Collections object. In the example below, a list of subjects is stored in the Student Java Bean class.

Student.java

```
package com.yawintutor;

import java.util.List;

public class Student {
	private int id;
	private String name;
	public List<String> subjects;

	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

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

	public List<String> getSubjects() {
		return subjects;
	}

	public void setSubjects(List<String> subjects) {
		this.subjects = subjects;
	}

}
```

StudentMain.java

```
package com.yawintutor;

import java.util.ArrayList;
import java.util.List;

public class StudentMain {
	public static void main(String[] args) {
		Student student = new Student();
```

```
                List<String> subjects = new ArrayList<String>();
                subjects.add("English");
                subjects.add("Science");
                subjects.add("Computer");

                // setting bean values
                student.setId(101);
                student.setName("Karthik");
                student.setSubjects(subjects);

                // getting bean value
                System.out.println("Student Id     : " + student.getId());
                System.out.println("Student name   : " + student.getName());

                List<String> subjectList = student.getSubjects();
                for (int i = 0; i < subjectList.size(); i++) {
                        System.out.println("Student subject " + (i + 1) + " : " + subjectList.get(i));
                }
        }
}
```

Output

```
Student Id     : 101
Student name   : Karthik
Student subject 1 : English
Student subject 2 : Science
Student subject 3 : Computer
```

Nested JavaBeans
Nested JavaBean is a Java Bean, containing one or more Java Beans. There are two javabeans called Student and Subject. The Student class has one or more Subject Java Beans.
Subject.java

```
package com.yawintutor;

public class Subject {
        private int code;
        private String name;

        public int getCode() {
                return code;
        }

        public void setCode(int code) {
                this.code = code;
        }

        public String getName() {
                return name;
```

```
          }

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

}
```

Student.java

```
package com.yawintutor;

import java.util.List;

public class Student {
          private int id;
          private String name;
          public List<Subject> subjects;

          public int getId() {
                    return id;
          }

          public void setId(int id) {
                    this.id = id;
          }

          public String getName() {
                    return name;
          }

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

          public List<Subject> getSubjects() {
                    return subjects;
          }

          public void setSubjects(List<Subject> subjects) {
                    this.subjects = subjects;
          }

}
```

StudentMain.java

```
package com.yawintutor;

import java.util.ArrayList;
import java.util.List;
```

```
public class StudentMain {
		public static void main(String[] args) {
				Student student = new Student();
				List<Subject> subjects = new ArrayList<Subject>();
				Subject subject1 = new Subject();
				subject1.setCode(210);
				subject1.setName("English");
				subjects.add(subject1);

				Subject subject2 = new Subject();
				subject2.setCode(220);
				subject2.setName("Science");
				subjects.add(subject2);

				Subject subject3 = new Subject();
				subject3.setCode(230);
				subject3.setName("Computer");
				subjects.add(subject3);

				// setting bean values
				student.setId(101);
				student.setName("Karthik");
				student.setSubjects(subjects);

				// getting bean value
				System.out.println("Student Id     : " + student.getId());
				System.out.println("Student name   : " + student.getName());

				List<Subject> subjectList = student.getSubjects();
				for (int i = 0; i < subjectList.size(); i++) {
						Subject subject = subjectList.get(i);
						System.out.println("Student subject " + (i + 1) + " : " + subject.getCode()
+ " : " + subject.getName());
				}
		}
}
```

Output

```
Student Id     : 101
Student name   : Karthik
Student subject 1 : 210 : English
Student subject 2 : 220 : Science
Student subject 3 : 230 : Computer
```