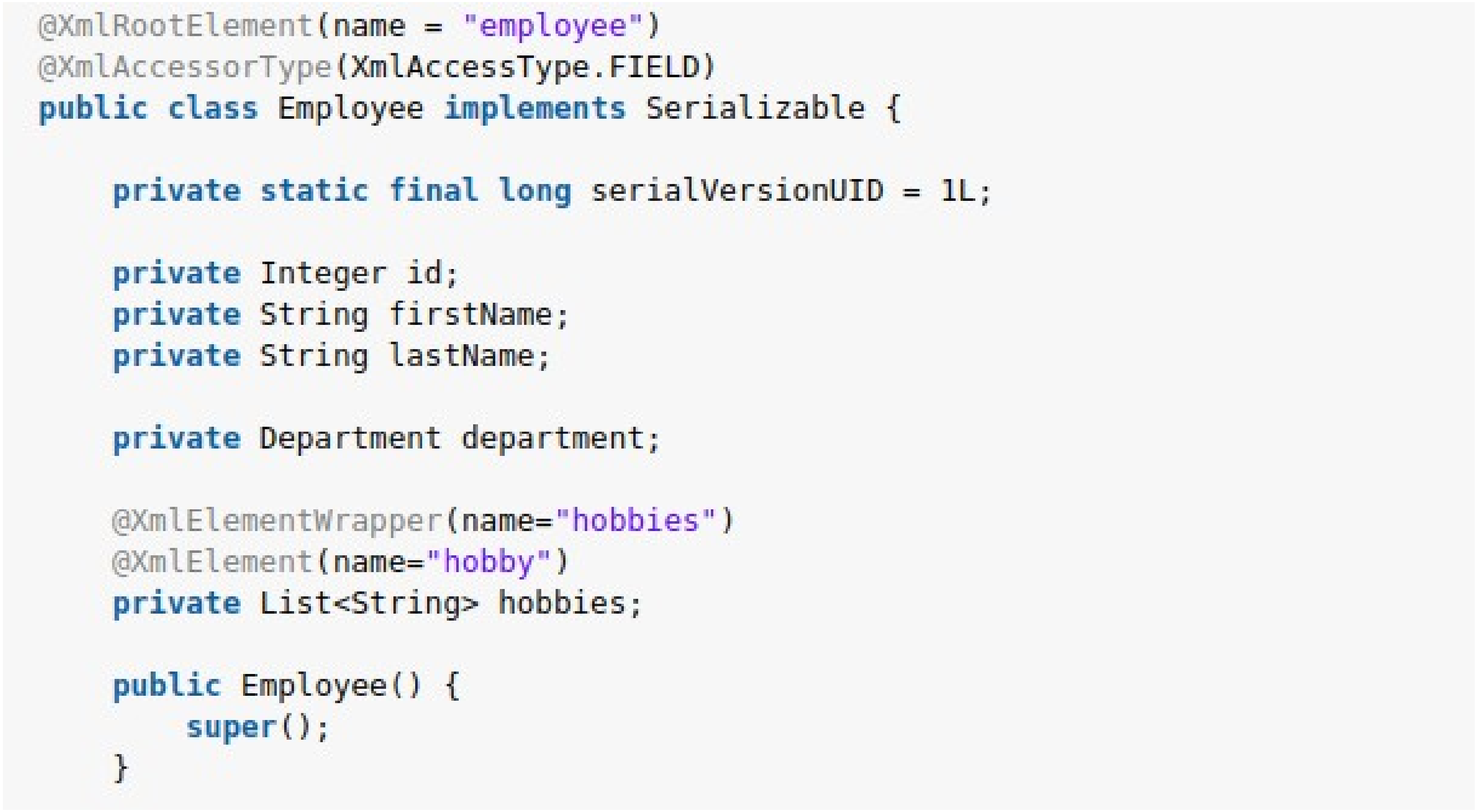

```java
@XmlRootElement(name = "employee")
@XmlAccessorType(XmlAccessType.FIELD)
public class Employee implements Serializable {

    private static final long serialVersionUID = 1L;

    private Integer id;
    private String firstName;
    private String lastName;

    private Department department;

    @XmlElementWrapper(name="hobbies")
    @XmlElement(name="hobby")
    private List<String> hobbies;

    public Employee() {
        super();
    }
```

```
public static void main(String[] args)
{
    Employee employee = new Employee(1, "Lokesh", "Gupta", new Department(101, "IT")

    employee.setHobbies(Arrays.asList("Swimming","Playing", "Karate"));

    jaxbObjectToXML(employee);
}

private static void jaxbObjectToXML(Employee employee)
{
    try {
        JAXBContext jaxbContext = JAXBContext.newInstance(Employee.class);
        Marshaller jaxbMarshaller = jaxbContext.createMarshaller();

        jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);

        //Print XML String to Console
        jaxbMarshaller.marshal(employee, new File("employee.xml"));

    } catch (JAXBException e) {
        e.printStackTrace();
    }
}
```

1) List of JAXB Annotations

ANNOTATION	SCOPE	DESCRIPTION
`@XmlRootElement`	Class, Enum	Defines the XML root element. Root Java classes need to be registered with the JAXB context when it is created.
`@XmlAccessorType`	Package, Class	Defines the fields and properties of your Java classes that the JAXB engine uses for binding. It has four values: `PUBLIC_MEMBER`, `FIELD`, `PROPERTY` and `NONE`.
`@XmlAccessorOrder`	Package, Class	Defines the sequential order of the children.
`@XmlType`	Class, Enum	Maps a Java class to a schema type. It defines the type name and order of its children.
`@XmlElement`	Field	Maps a field or property to an XML element
`@XmlAttribute`	Field	Maps a field or property to an XML attribute

`@XmlAccessorOrder`	Package, Class	Defines the sequential order of the children.
`@XmlType`	Class, Enum	Maps a Java class to a schema type. It defines the type name and order of its children.
`@XmlElement`	Field	Maps a field or property to an XML element
`@XmlAttribute`	Field	Maps a field or property to an XML attribute
`@XmlTransient`	Field	Prevents mapping a field or property to the XML Schema
`@XmlValue`	Field	Maps a field or property to the text value on an XML tag.
`@XmlList`	Field, Parameter	Maps a collection to a list of values separated by space.
`@XmlElementWrapper`	Field	Maps a Java collection to an XML wrapped collection

1.1) @XmlRootElement

This maps a class or an enum type to an XML root element. When a top-level class or an enum type is annotated with the `@XmlRootElement` annotation, then its value is represented as XML element in an XML document.

Employee.java

```java
@XmlRootElement(name = "employee")
@XmlAccessorType(XmlAccessType.PROPERTY)
public class Employee implements Serializable
{
    //More code
}
```

Above will result into:

employee.xml

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<employee>
    //....
</employee>
```

1.2) @XmlAccessorType

It defines the fields or properties of your Java classes that the JAXB engine uses for including into generated XML. It has four possible values.

- `FIELD` – Every non static, non transient field in a JAXB-bound class will be automatically bound to XML, unless annotated by `XmlTransient`.
- `NONE` – None of the fields or properties is bound to XML unless they are specifically annotated with some of the JAXB annotations.
- `PROPERTY` – Every getter/setter pair in a JAXB-bound class will be automatically bound to XML, unless annotated by `XmlTransient`.
- `PUBLIC_MEMBER` – Every public getter/setter pair and every public field will be automatically bound to XML, unless annotated by `XmlTransient`.
- Default value is `PUBLIC_MEMBER`.

Employee.java

```java
@XmlRootElement(name = "employee")
@XmlAccessorType(XmlAccessType.FIELD)
public class Employee implements Serializable
{
    private Integer id;
    private String firstName;
    private String lastName;
}
```

- `PUBLIC_MEMBER` – Every public getter/setter pair and every public field will be automatically bound to XML, unless annotated by `XmlTransient`.
- Default value is `PUBLIC_MEMBER`.

Employee.java

```java
@XmlRootElement(name = "employee")
@XmlAccessorType(XmlAccessType.FIELD)
public class Employee implements Serializable
{
    private Integer id;
    private String firstName;
    private String lastName;
}
```

Above will result into:

employee.xml

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<employee>
    <firstName>Lokesh</firstName>
    <id>1</id>
    <lastName>Gupta</lastName>
</employee>
```

1.3) @XmlAccessorOrder

Controls the ordering of fields and properties in a class. You can have predefined values `ALPHABETICAL` or `UNDEFINED`.

Employee.java

```java
@XmlRootElement(name = "employee")
@XmlAccessorType(XmlAccessType.FIELD)
public class Employee implements Serializable
{
    private Integer id;
    private String firstName;
    private String lastName;
    private Department department;
}
```

Above will result into:

employee.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<employee>
   <department>
      <id>101</id>
      <name>IT</name>
   </department>
   <firstName>Lokesh</firstName>
   <id>1</id>
   <lastName>Gupta</lastName>
</employee>
```

1.4) @XmlType

It maps a Java class or enum type to a schema type. It defines the type name, namespace and order of its children. It is used to match the element in the schema to element in the model.

Employee.java

```
@XmlRootElement(name = "employee")
@XmlType(propOrder={"id", "firstName" , "lastName", "department" })
public class Employee implements Serializable
{
    private Integer id;
    private String firstName;
    private String lastName;
    private Department department;
}
```

1.5) @XmlElement

Maps a JavaBean property to an XML element derived from property name.

Employee.java

```java
@XmlRootElement(name = "employee")
public class Employee implements Serializable
{
    @XmlElement(name=employeeId)
    private Integer id;

    @XmlElement
    private String firstName;

    private String lastName;
    private Department department;
}
```

Above will result into:

employee.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<employee>
    <employeeId>1</employeeId>
    <firstName>Lokesh</firstName>
</employee>
```

1.6) @XmlAttribute

Maps a JavaBean property to an XML attribute.

Employee.java

```java
@XmlRootElement(name = "employee")
public class Employee implements Serializable
{
    @XmlAttribute
    private Integer id;

    private String firstName;
    private String lastName;
    private Department department;
}
```

Above will result into:

employee.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<employee id="1">
   <department>
      <id>101</id>
      <name>IT</name>
   </department>
   <firstName>Lokesh</firstName>
   <lastName>Gupta</lastName>
</employee>
```

1.7) @XmlTransient

Prevents the mapping of a JavaBean property/type to XML representation. When placed on a class, it indicates that the class shouldn't be mapped to XML by itself. Properties on such class will be mapped to XML along with its derived classes as if the class is inlined.

`@XmlTransient` is mutually exclusive with all other JAXB defined annotations.

Employee.java

```java
@XmlRootElement(name = "employee")
@XmlAccessorType(XmlAccessType.FIELD)
public class Employee implements Serializable
{
    @XmlTransient
    private Integer id;

    private String firstName;
    private String lastName;
    private Department department;
}
```

Above will result into:

id field is not serialized

```xml
<?xml version="1.0" encoding="UTF-8"?>
<employee>
   <firstName>Lokesh</firstName>
   <lastName>Gupta</lastName>
   <department>
      <id>101</id>
      <name>IT</name>
   </department>
</employee>
```

1.9) @XmlList

Used to map a property to a list simple type. It allows multiple values to be represented as **whitespace-separated tokens** in a single element.

```
Employee.java
```

```java
@XmlRootElement(name = "employee")
@XmlAccessorType(XmlAccessType.FIELD)
public class Employee implements Serializable
{
    private List<String> hobbies;
}

//

<?xml version="1.0" encoding="UTF-8"?>
<employee>
    <hobbies>Swimming</hobbies>
    <hobbies>Playing</hobbies>
    <hobbies>Karate</hobbies>
</employee>
```

After using `@XmlList`, observe the output.

```
Employee.java
```

```
@XmlRootElement(name = "employee")
@XmlAccessorType(XmlAccessType.FIELD)
public class Employee implements Serializable
{
    @XmlList
    private List<String> hobbies;
}

//

<?xml version="1.0" encoding="UTF-8"?>
<employee>
    <hobbies>Swimming Playing Karate</hobbies>
</employee>
```

1.10) @XmlElementWrapper

Generates a wrapper element around XML representation. This is primarily intended to be used to produce a wrapper XML element around collections. So, it must be used with collection property.

Employee.java

```java
@XmlRootElement(name = "employee")
@XmlAccessorType(XmlAccessType.FIELD)
public class Employee implements Serializable
{
    @XmlElementWrapper(name="hobbies")
    @XmlElement(name="hobby")
    private List<String> hobbies;
}
```

Above will result into:

id field is not serialized

```xml
<?xml version="1.0" encoding="UTF-8"?>
<employee>
    <hobbies>
        <hobby>Swimming</hobby>
        <hobby>Playing</hobby>
        <hobby>Karate</hobby>
    </hobbies>
</employee>
```