

```
import java.util.*;
```

```
/**
```

```
 * Class billing_page
```

```
 */
```

```
public class billing_page {
```

```
    //
```

```
    // Fields
```

```
    //
```

```
    //
```

```
    // Constructors
```

```
    //
```

```
    public billing_page () { };
```

```
    //
```

```
    // Methods
```

```
    //
```

```
    //
```

```
    // Accessor methods
```

```
    //
```

```
    //
```

```
    // Other methods
```

```
    //
```

```
}
```

```
/**
```

```
 * Class blood
```

```
 */
```

```
public class blood {
```

```
    //
```

```
    // Fields
```

```
    //
```

```
    private Integer id;
```

```
    private Long donar_id;
```

```
    private Short mobile;
```

```
    private Double patiend_id;
```

```
    //
```

```
    // Constructors
```

```
    //
```

```
    public blood () { };
```

```
    //
```

```
    // Methods
```

```
    //
```

```
    //
```

```
    // Accessor methods
```

```
    //
```

```
/**
 * Set the value of id
 * @param newVar the new value of id
 */
private void setId (Integer newVar) {
    id = newVar;
}
```

```
/**
 * Get the value of id
 * @return the value of id
 */
private Integer getId () {
    return id;
}
```

```
/**
 * Set the value of donar_id
 * @param newVar the new value of donar_id
 */
private void setDonar_id (Long newVar) {
    donar_id = newVar;
}
```

```
/**
 * Get the value of donar_id
 * @return the value of donar_id
 */
private Long getDonar_id () {
    return donar_id;
}
```

```
}
```

```
/**
```

```
 * Set the value of mobile
```

```
 * @param newVar the new value of mobile
```

```
 */
```

```
private void setMobile (Short newVar) {
```

```
    mobile = newVar;
```

```
}
```

```
/**
```

```
 * Get the value of mobile
```

```
 * @return the value of mobile
```

```
 */
```

```
private Short getMobile () {
```

```
    return mobile;
```

```
}
```

```
/**
```

```
 * Set the value of patiend_id
```

```
 * @param newVar the new value of patiend_id
```

```
 */
```

```
private void setPatiend_id (Double newVar) {
```

```
    patiend_id = newVar;
```

```
}
```

```
/**
```

```
 * Get the value of patiend_id
```

```
 * @return the value of patiend_id
```

```
 */
```

```
private Double getPatiend_id () {
```

```
    return patiend_id;  
}
```

```
//  
// Other methods  
//
```

```
/**  
 * @return    String  
 */  
public String add()  
{  
}
```

```
/**  
 * @return    Short  
 */  
public Short edit()  
{  
}
```

```
/**  
 * @return    Long  
 */  
public Long delete()  
{  
}
```

The screenshot displays the Umbrello UML Modeller application. The main workspace contains a Use Case Diagram for a blood bank system. The diagram features two actors: 'super admin' and 'system user'. 'super admin' is connected to 'manage user', 'manage blood', 'manage blood group', 'login', 'update profile', 'change password', 'logout', 'manage blood cell', 'manage donor', and 'manage stock'. 'system user' is connected to 'manage blood cell', 'manage donor', and 'manage stock'. Additionally, there is a vertical sequence of use cases on the right: 'search blood' connected to 'view inventory', which is connected to an unnamed actor, which is connected to 'donor', 'enquiry', 'request', and 'make payment'. The left sidebar includes a 'Stereotypes' panel with a table of stereotypes (Name, Usage) and a 'Tree View' panel showing the project structure. The bottom status bar indicates 'Press Ctrl with left mouse click to delete a point'.

Name	Usage
folder	1
datatype	24
interface	1

Class Name	Properties
umlwidgets	<input type="checkbox"/>
umbrello	<input type="checkbox"/>
cpparser	<input type="checkbox"/>
dialogs	<input checked="" type="checkbox"/>
codeimport	<input checked="" type="checkbox"/>
refactoring	<input checked="" type="checkbox"/>
umlmodel	<input checked="" type="checkbox"/>

Documentation: Command history | Log

Press Ctrl with left mouse click to delete a point

99% Fit: 100%

