

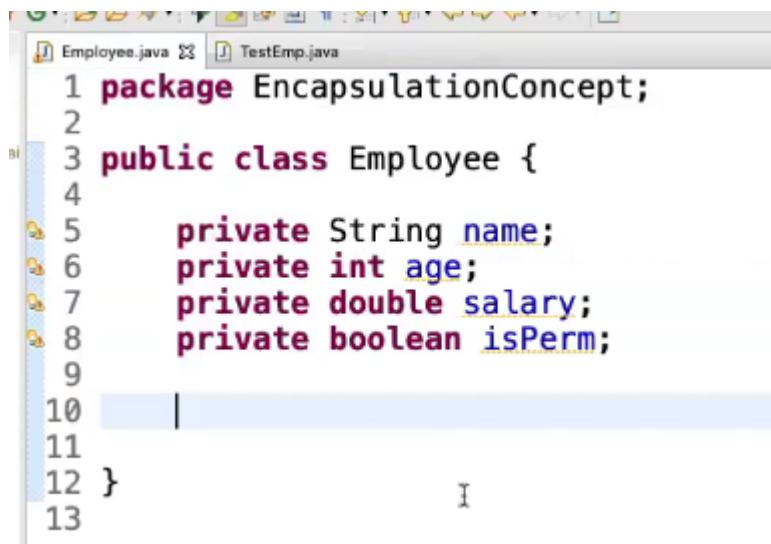
## Encapsulation-

No unnecessary things shown to user.

Data security.

We can keep all the variables as private.

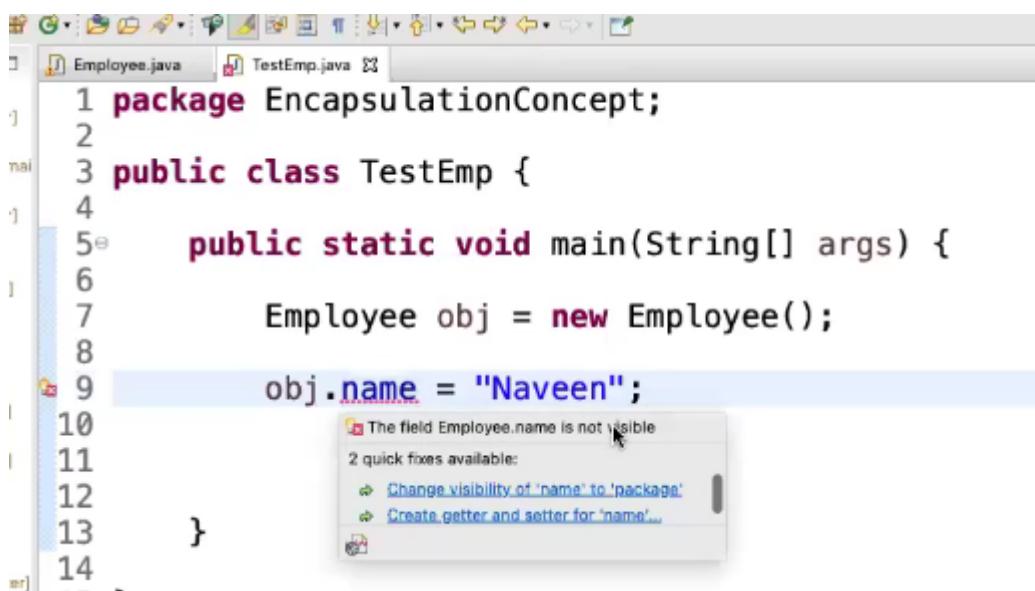
Methods will be public.



```

1 package EncapsulationConcept;
2
3 public class Employee {
4
5     private String name;
6     private int age;
7     private double salary;
8     private boolean isPerm;
9
10    |
11
12 }
13
  
```

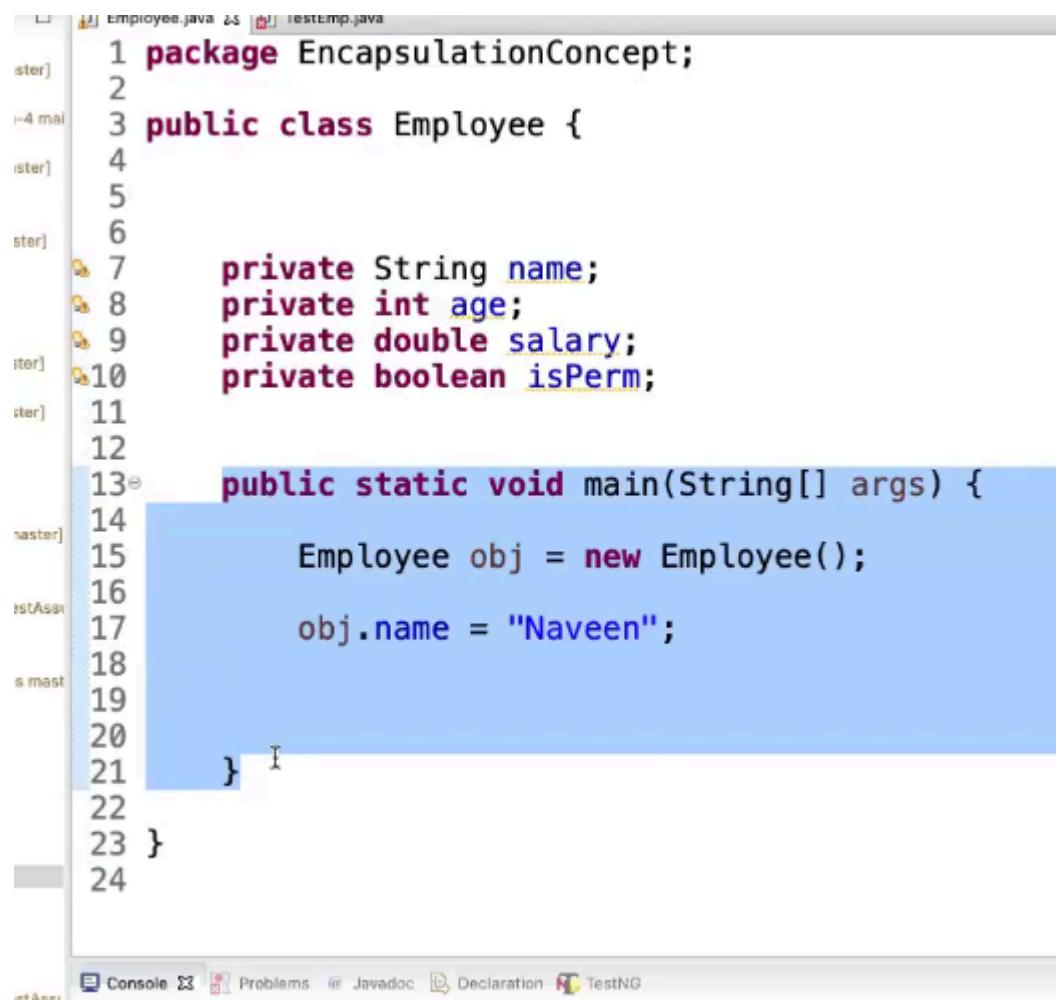
Cant access private from other class-



```

1 package EncapsulationConcept;
2
3 public class TestEmp {
4
5     public static void main(String[] args) {
6
7         Employee obj = new Employee();
8
9         obj.name = "Naveen";
10
11
12     }
13
14
  
```

Within class everything allowed-



```

1 package EncapsulationConcept;
2
3 public class Employee {
4
5
6
7     private String name;
8     private int age;
9     private double salary;
10    private boolean isPerm;
11
12
13    public static void main(String[] args) {
14
15        Employee obj = new Employee();
16
17        obj.name = "Naveen";
18
19
20    }
21
22
23 }
24

```

The screenshot shows a Java code editor with the file 'Employee.java' open. The code defines a class 'Employee' with private fields for name, age, salary, and isPerm. A main method is present. The line 'obj.name = "Naveen";' is highlighted with a blue background. The bottom of the screen shows the Eclipse interface with tabs for Console, Problems, Javadoc, Declaration, and TestNG.

## Getter and setter-

```

9
10
11     //public methods: getter/setter
12
13    public void setName(String name) {
14        this.name = name;
15    }
16
17    public String getName() {
18        return this.name;| +
19    }
20
21
22

```

The screenshot shows an IDE interface with two tabs open: 'Employee.java' and 'TestEmp.java'. The 'Employee.java' tab contains a simple class definition:

```

1 package EncapsulationConcept;
2
3 public class Employee {
4
5     public void printName() {
6         System.out.println("Employee Name");
7     }
8
9 }

```

The 'TestEmp.java' tab contains the following code:

```

1 package EncapsulationConcept;
2
3 public class TestEmp {
4
5     public static void main(String[] args) {
6
7         Employee obj = new Employee();
8
9         obj.setName("Tom");
10        String n = obj.getName();
11        System.out.println(n);
12    }
13
14 }
15

```

In the bottom right corner, the 'Console' tab shows the output: 'Tom'.

## Getters and setters for all-

```

9
10 // public methods: getter/setter
11
12 public String getName() {
13     return name;
14 }
15
16 public void setName(String name) {
17     this.name = name;
18 }
19
20 public int getAge() {
21     return age;
22 }
23
24 public void setAge(int age) {
25     this.age = age;
26 }
27
28 public double getSalary() {
29     return salary;
30 }
31
32 public void setSalary(double salary) {
33     this.salary = salary;
34 }
35
36 public boolean isPerm() {
37     return isPerm;
38 }
39
40 public void setPerm(boolean isPerm) {
41     this.isPerm = isPerm;
42 }
43

```

this is test emp class-

```

12
13
14     obj.setAge(20);
15     System.out.println(obj.getAge());
16

```

20

Constructor created-

```

8     private double salary;
9     private boolean isPerm;
10
11    //public const..
12    public Employee(String name, int age, double salary, boolean isPerm) {
13        this.name = name;
14        this.age = age;
15        this.salary = salary;
16        this.isPerm = isPerm;
17    }
18
19    // public methods: getter/setter
20
21    //public const.. is also like setter
22    public Employee(String name, int age, double salary, boolean isPerm) {
23        this.name = name;
24        this.age = age;
25        this.salary = salary;
26        this.isPerm = isPerm;
27    }

```

test emp class-

```

16
17     Employee obj = new Employee("Pooja", 20, 12.33, true);
18
19     System.out.println(obj.getName() + " " + obj.getAge() + " " + obj.getSalary() + " " + obj.isPerm());
20
21

```

```

Console 3 Problems @ Java
<terminated> TestEmp (2) [Java Application]
Pooja 20 12.33 true

```

Setter can be done by constructor so not mandatory to write setters individually.

Setter used for updating values-

## test emp class-

```

21
22     obj.setAge(21);
23     obj.setSalary(25);
24
25     System.out.println(obj.getName() + " " + obj.getAge() + " " + obj.getSalary() + " " + obj.isPerm());
26
27 }
28
29 }
30

```

Pooja 21 25.0 true

## Another object-

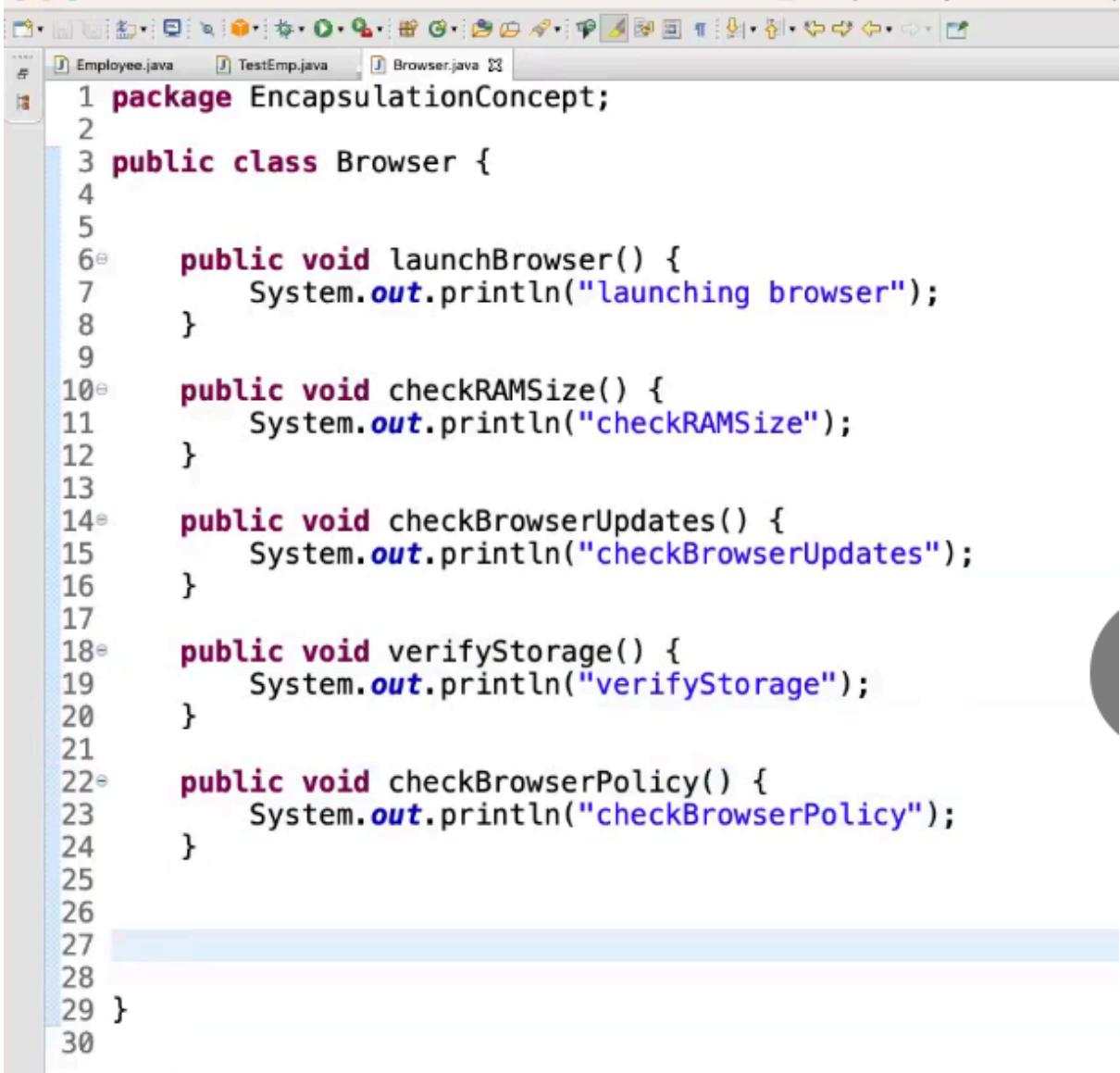
```

31
32     // 
33     Employee obj1 = new Employee("Veena", 30, 22.33, false);
34     System.out.println(obj1.getName() + " " + obj1.getAge() + " " + obj1.getSalary() + " " + obj1.isPerm());
35
36     obj1.setPerm(true);
37
38     System.out.println(obj1.getName() + " " + obj1.getAge() + " " + obj1.getSalary() + " " + obj1.isPerm());

```

Veena 30 22.33 false  
Veena 30 22.33 true

## Another class-

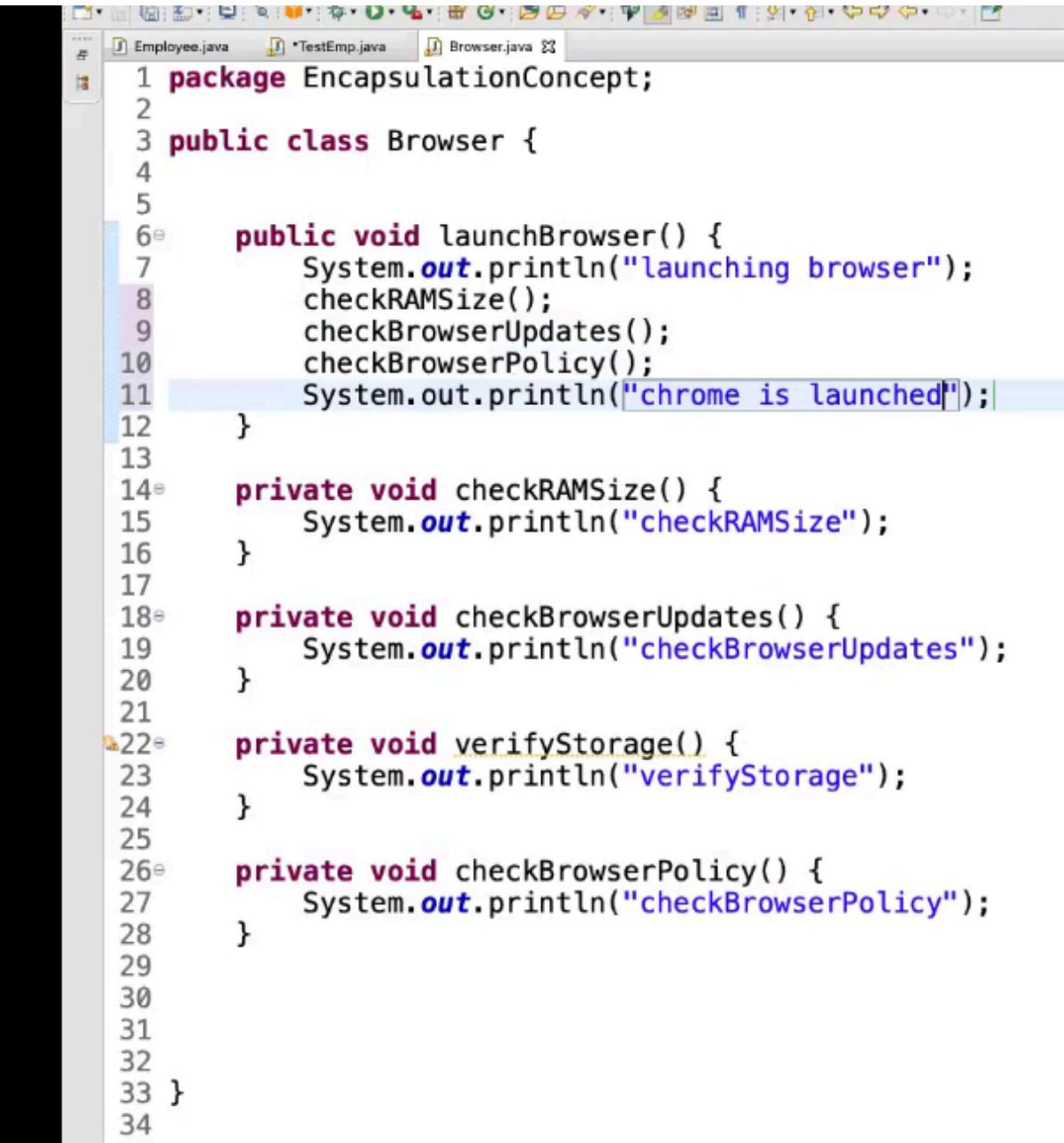


The screenshot shows a Java IDE interface with a toolbar at the top and three tabs open: Employee.java, TestEmp.java, and Browser.java. The Browser.java tab is active, displaying the following code:

```
1 package EncapsulationConcept;
2
3 public class Browser {
4
5
6    public void launchBrowser() {
7        System.out.println("launching browser");
8    }
9
10   public void checkRAMSize() {
11       System.out.println("checkRAMSize");
12   }
13
14   public void checkBrowserUpdates() {
15       System.out.println("checkBrowserUpdates");
16   }
17
18   public void verifyStorage() {
19       System.out.println("verifyStorage");
20   }
21
22   public void checkBrowserPolicy() {
23       System.out.println("checkBrowserPolicy");
24   }
25
26
27
28
29 }
30
```

Bad design. Why to keep all methods as public which is not useful.

Encapsulate-



The screenshot shows a Java IDE interface with three tabs at the top: Employee.java, TestEmp.java, and Browser.java. The Browser.java tab is active, displaying the following code:

```
1 package EncapsulationConcept;
2
3 public class Browser {
4
5
6     public void launchBrowser() {
7         System.out.println("launching browser");
8         checkRAMSize();
9         checkBrowserUpdates();
10        checkBrowserPolicy();
11        System.out.println("chrome is launched");
12    }
13
14    private void checkRAMSize() {
15        System.out.println("checkRAMSize");
16    }
17
18    private void checkBrowserUpdates() {
19        System.out.println("checkBrowserUpdates");
20    }
21
22    private void verifyStorage() {
23        System.out.println("verifyStorage");
24    }
25
26    private void checkBrowserPolicy() {
27        System.out.println("checkBrowserPolicy");
28    }
29
30
31
32
33 }
34
```

Below the code editor, there is a code completion or search panel with the following suggestions:

```
39
40
41     //browser:
42     Browser br = new Browser();
43     br.launchBrowser();
44
```

```
4  
5  public void launchBrowser() {  
6      System.out.println("launching browser");  
7      checkRAMSize();  
8      checkBrowserUpdates();  
9      checkBrowserPolicy();  
10     verifyStorage();  
11     System.out.println("chrome is launched");  
12 }
```

Encapsulation for both methods and variables.

```
launching browser  
checkRAMSize  
obj.checkBrowserUpdates  
checkBrowserPolicy  
verifyStorage  
I chrome is launched
```

paste browser1

```
1 package com.day16;
2
3 public class browser1 {
4
5     //encapsulation can be for methods and variables.
6     //good example of encapsulation.
7
8     public void launchbrowser() {
9         System.out.println("launch browser");
10        checkramsize();
11        checkbrowserupdates();
12        int b1= verifystorage();
13        System.out.println(b1);
14        checkbrowserpolicy();
15        System.out.println("end of the public method");
16    }
17
18    private void checkramsize() {
19        System.out.println("check ram size");
20    }
21
22    private void checkbrowserupdates() {
23        System.out.println("check for browser updates");
24    }
25
26    //we can have return type also.
27    private int verifystorage() {
28        System.out.println("verify storage");
29        return 10;
30    }
31
32    private void checkbrowserpolicy() {
33        System.out.println("check browser policy");
34    }
35}
```

```
33     System.out.println("check browser policy");
34 }
35
36 public static void main(String[] args) {
37
38     browser1 b1=new browser1();
39     b1.launchbrowser();
40 }
41
42 }
43
44 //launch browser
45 //check ram size
46 //check for browser updates
47 //verify storage
48 //10
49 //check browser policy
50 //end of the public method
51
52
```

paste employee2 and test employee2

The screenshot shows a Java code editor interface with two tabs at the top: "employee2.java" and "testemployee2.java". The "employee2.java" tab is active, displaying the following Java code:

```
1 package com.day16;
2
3 public class employee2 {
4
5     private String name;
6     private int age;
7     private double salary;
8     private boolean isPermanent;
9
10    //constructor with all fields.
11    //constructor is also like setter.
12    public employee2(String name, int age, double salary, boolean isPermanent) {
13        this.name=name;
14        this.age=age;
15        this.salary=salary;
16        this.isPermanent=isPermanent;
17    }
18
19    //setter method for name.
20    public void setName(String name) {
21        this.name=name;
22    }
23
24    //getter method for name.
25    public String getName() {
26        return this.name;
27    }
28
29    //getter for all other variables.
30
31    public int getAge() {
32        return this.age;
33    }
34}
```

```

32     return this.age;
33 }
34
35 //this not mandatory for get method.
36 public double getSalary() {
37     return salary;
38 }
39
40 public boolean getIsPermanent() {
41     return isPermanent;
42 }
43
44 //setter method for all other variables.
45
46 public void setAge(int age) {
47     this.age=age;
48 }
49
50 public void setSalary(double salary) {
51     this.salary=salary;
52 }
53
54 public void setIsPermanent(boolean isPermanent) {
55     this.isPermanent=isPermanent;
56 }
57
58
59 }
60

```

testemployee2.java

```

1 package com.day16;
2
3 public class testemployee2 {
4
5     public static void main(String[] args) {
6
7         employee2 e1=new employee2("karan", 40, 50000.0, true);
8         System.out.println(e1.getAge() + " " + e1.getName() + " " + e1.getSalary() + " " + e1.getIsPermanent());
9
10        e1.setSalary(332443.234324);
11        System.out.println(e1.getAge() + " " + e1.getName() + " " + e1.getSalary() + " " + e1.getIsPermanent());
12
13        e1.setIsPermanent(false);
14        System.out.println(e1.getAge() + " " + e1.getName() + " " + e1.getSalary() + " " + e1.getIsPermanent());
15    }
16
17 }
18
19 //40 karan 50000.0 true
20 //40 karan 332443.234324 true
21 //40 karan 332443.234324 false
22
23
24

```

## Another example-

The screenshot shows a Java development environment with two tabs open: `LoginPage.java` and `TestEmp.java`. The `LoginPage.java` tab is active, displaying the following code:

```
1 package EncapsulationConcept;
2
3 public class LoginPage {
4
5     private String userName;
6     private String password;
7
8     public String getUserName() {
9         return userName;
10    }
11
12    public void setUserName(String userName) {
13        this.userName = userName;
14    }
15
16    public String getPassword() {
17        return password;
18    }
19
20    public void setPassword(String password) {
21        this.password = password;
22    }
23
24 }
25 |
```

The `TestEmp.java` tab is partially visible below it, showing the following code:

```
44
45     //login:
46     LoginPage lp1 = new LoginPage();
47     lp1.setUserName("naveen@gmail.com");
48     lp1.setPassword("naveen@123");
49
50 |
```

paste login1

The screenshot shows a Java code editor window with the file `Login1.java` open. The code defines a class `Login1` with private fields `username` and `password`, and public methods to get and set these values. It also contains a `main` method that creates an instance of `Login1`, sets the username and password, and prints them to the console.

```
1 package com.day16;
2
3 public class Login1 {
4
5     private String username;
6     private String password;
7
8
9     public String getUsername() {
10         return username;
11     }
12
13     public void setUsername(String username) {
14         this.username = username;
15     }
16
17     public String getPassword() {
18         return password;
19     }
20
21     public void setPassword(String password) {
22         this.password = password;
23     }
24
25     public static void main(String[] args) {
26         Login1 l1=new Login1();
27         l1.setUsername("admin");
28         l1.setPassword("admin@##$23213");
29         String s1=l1.getPassword();
30         String s2=l1.getUsername();
31
32         System.out.println(s1);
33         System.out.println(s2);
34     }
35 }
```

```
33     System.out.println(s2);
34 }
35
36 }
37
38 //admin@##$23213
39 //admin
40
41
```



The screenshot shows a Java code editor window titled "Login1.java". The code defines a class "Login1" with private fields "username" and "password", and public methods to get and set these values. It also contains a main method that creates an instance of "Login1", sets its username and password, and then prints them out. The code is numbered from 1 to 41.

```
1 package com.day16;
2
3 public class Login1 {
4
5     private String username;
6     private String password;
7
8
9     public String getUsername() {
10         return username;
11     }
12
13     public void setUsername(String username) {
14         this.username = username;
15     }
16
17     public String getPassword() {
18         return password;
19     }
20
21     public void setPassword(String password) {
22         this.password = password;
23     }
24
25     public static void main(String[] args) {
26         Login1 l1=new Login1();
27         l1.setUsername("admin");
28         l1.setPassword("admin@##$23213");
29         String s1=l1.getPassword();
30         String s2=l1.getUsername();
31
32         System.out.println(s1);
33         System.out.println(s2);
34     }
35
36 }
37
38 //admin@##$23213
39 //admin
40
41
```

Add constructor-

```

7
8  public LoginPage(String userName, String password) {
9      this.userName = userName;
10     this.password = password;
11 }

```

## Add method-

```

29
30     public void doLogin(String un, String pwd) {
31         System.out.println("enter username: " + un);
32         System.out.println("enter password: " + pwd);
33         System.out.println("click on login button");
34         System.out.println("user is logged in");
35     }

45     //login:
46     //POST - create credes
47     LoginPage lp1 = new LoginPage("naveen@gmail.com", "naveen@123");
48
49     lp1.doLogin(lp1.getUserName(), lp1.getPassword());
50

```

```

enter username: naveen@gmail.com
enter password: naveen@123
click on login button
user is logged in

```

## Modified code-

```

27
28
29  public void doLogin() {
30     System.out.println("enter username: " + getUserName());
31     System.out.println("enter password: " + getPassword());
32     System.out.println("click on login button");
33     System.out.println("user is logged in");
34 }
35

```

```
44
45     //login:
46     //POST - create credes
47     LoginPage lp1 = new LoginPage("naveen@gmail.com", "naveen@123");
48
49     lp1.doLogin();
50
```

enter username: naveen@gmail.com  
enter password: naveen@123  
click on login button  
user is logged in



## Get user name and password-

```
50
51     System.out.println(lp1.getUserName() + " " + lp1.getPassword());
52

user is logged in
naveen@gmail.com naveen@123
```

## Set the values—

```
53     lp1.setPassword("naveen@9090");
54
55     System.out.println(lp1.getUserName() + " " + lp1.getPassword());
56 
```

## Create another object–

```
56
57     System.out.println("-----");
58     //
59     LoginPage lp2 = new LoginPage("piyush@gmail.com", "pityus@123");
60     lp2.doLogin();
61
62 }
```

-----  
enter username: piyush@gmail.com  
enter password: pityus@123  
click on login button  
+ user is logged in

Another code-

## paste bank1

The screenshot shows a Java code editor window with the file `bank1.java` open. The code defines a class `bank1` with private fields `name`, `age`, `adhar`, and `phone`. It includes a constructor that takes `String name`, `int age`, `String adhar`, and `String phone` and initializes the respective fields. It also includes three getter methods: `getName()`, `getAge()`, and `getAdhar()`.

```
1 package com.day16;
2
3 public class bank1 {
4     private String name;
5     private int age;
6     private String adhar;
7     private String phone;
8
9     // Constructor
10    public bank1(String name, int age, String adhar, String phone) {
11        this.name = name;
12        this.age = age;
13        this.adhar = adhar;
14        this.phone = phone;
15    }
16
17    public bank1() {
18        // TODO Auto-generated constructor stub
19    }
20
21    // Getter
22    public String getName() {
23        return name;
24    }
25
26    public int getAge() {
27        return age;
28    }
29
30    public String getAdhar() {
31        return adhar;
32    }
33
34    public String getPhone() {
35        return phone;
36    }
37}
```

```
35     return phone;
36 }
37
38 // Setter
39 public void setName(String name) {
40     this.name = name;
41 }
42
43 //we can write any damn code inside getters and setters.
44 public void setAge(int age) {
45     if(age>=16) {
46         this.age=age;
47         System.out.println("hello olders");
48     }
49     else {
50         System.out.println("grow up to 16 years");
51     }
52 }
53
54 public void setAdhar(String adhar) {
55     this.adhar = adhar;
56 }
57
58 public void setPhone(String phone) {
59     this.phone = phone;
60 }
61 }
62 }
```

paste testbank1

```

1 package com.day16;
2
3 public class testbank1 {
4
5     public static void main(String[] args) {
6
7         bank1 b1=new bank1();
8         b1.setAge(15);
9         b1.setName("karan");
10
11         //print name and age in one line.
12
13         System.out.println(b1.getName()+" "+b1.getAge());
14
15         //create another object.
16         //different age greater than 16 years.
17         //different name.
18
19         bank1 b2=new bank1();
20         b2.setAge(20);
21         b2.setName("tiger");
22
23         System.out.println(b2.getName()+" "+b2.getAge());
24     }
25
26 }
27
28 //grow up to 16 years
29 //karan 0
30 //hello olders
31 //tiger 20
32
33

```

## Conditions in getters and setters allowed-

```

21
22     public void setAge(int age) {
23         if (age >= 15) {
24             this.age = age;
25         }
26     }

```

Change age to 16- covered above.

Note-

Keep all initial checks in setter.

Final variable no relation to encapsulation-

```

Employee.java TestEmp.java Browser.java LoginPage.java Bank.java 23 TestBank.java
1 package EncapsulationConcept;
2
3 public class Bank {
4
5     private final String name = "naveen";
6     private int age;// age>=15
7     private String aadharNumber;
8     private String phone;
9
10    public String getName() {
11        return name;
12    }
13
14    public void setName(String name) {
15        this.name = name;
16    }
17
18    public int getAge() {
19        return age;
20    }
21

```

//The final field bank2.name cannot be assigned

[paste bank2](#)

```
1 package com.day16;
2
3 public class bank2 {
4
5     //create final variable but dont give value.
6     private final String name;
7     private int age;
8     private String adhar;
9     private String phone;
10
11    // Constructor
12    @Override
13    public bank2(String name, int age, String adhar, String phone) {
14        this.name = name; //The final field bank2.name cannot be assigned
15        this.age = age;
16        this.adhar = adhar;
17        this.phone = phone;
18    }
19
20    @Deprecated
21    public bank2() { //The blank final field name may not have been initialized
22        // TODO Auto-generated constructor stub
23    }
24
25    // Getter
26    public String getName() {
27        return name;
28    }
29
30    public int getAge() {
31        return age;
32    }
33
34    public String getAdhar() {
35        return adhar;
36    }
37}
```

```
33         return adhar;
34     }
35
36     public String getPhone() {
37         return phone;
38     }
39
40     // Setter
41     public void setName(String name) {
42         this.name = name; //The final field bank2.name cannot be assigned
43     }
44
45     //we can write any damn code inside getters and setters.
46     public void setAge(int age) {
47         if(age>=16) {
48             this.age=age;
49             System.out.println("hello olders");
50         }
51         else {
52             System.out.println("grow up to 16 years");
53         }
54     }
55
56     public void setAdhar(String adhar) {
57         this.adhar = adhar;
58     }
59
60     public void setPhone(String phone) {
61         this.phone = phone;
62     }
63 }
64 }
```

paste bank3

The screenshot shows a Java code editor window with an orange border. At the top, there are two tabs: "bank2.java" and "bank3.java X". The "bank3.java" tab is active. The code editor displays the following Java code:

```
1 package com.day16;
2
3 public class bank3 {
4
5     //create final variable but give value.
6     //see the errors.
7     private final String name="gorilla";
8     private int age;
9     private String adhar;
10    private String phone;
11
12    // Constructor
13    public bank3(String name, int age, String adhar, String phone) {
14        this.name = name; //The final field bank2.name cannot be assigned
15        this.age = age;
16        this.adhar = adhar;
17        this.phone = phone;
18    }
19
20    // Getter
21    public String getName() {
22        return name;
23    }
24
25    public int getAge() {
26        return age;
27    }
28
29    public String getAdhar() {
30        return adhar;
31    }
32
33    public String getPhone() {
34        return phone;
35    }
36}
```

The code includes several annotations: line 14 has a warning about assigning to a final field; line 21 has a TODO comment; and lines 24-35 are empty bodies for getters.

```
34     return adhar;
35 }
36
37 public String getPhone() {
38     return phone;
39 }
40
41 // Setter
42 public void setName(String name) {
43     this.name = name; //The final field bank2.name cannot be assigned
44 }
45
46 //we can write any damn code inside getters and setters.
47 public void setAge(int age) {
48     if(age>=16) {
49         this.age=age;
50         System.out.println("hello olders");
51     }
52     else {
53         System.out.println("grow up to 16 years");
54     }
55 }
56
57 public void setAdhar(String adhar) {
58     this.adhar = adhar;
59 }
60
61 public void setPhone(String phone) {
62     this.phone = phone;
63 }
64 }
65 }
```

```
1 package com.day16;
2
3 public class bank3 {
4
5     //create final variable but give value.
6     //see the errors.
7     private final String name="gorilla";
8     private int age;
9     private String adhar;
10    private String phone;
11
12    // Constructor
13    public bank3(String name, int age, String adhar, String phone) {
14        this.name = name; //The final field bank2.name cannot be assigned
15        this.age = age;
16        this.adhar = adhar;
17        this.phone = phone;
18    }
19
20    public bank3() {
21        // TODO Auto-generated constructor stub
22    }
23
24    // Getter
25    public String getName() {
26        return name;
27    }
28
29    public int getAge() {
30        return age;
31    }
32
33    public String getAdhar() {
34        return adhar;
35    }
36
37    public String getPhone() {
38        return phone;
39    }
40
41    // Setter
42    public void setName(String name) {
43        this.name = name; //The final field bank2.name cannot be assigned
44    }
45
46    //we can write any damn code inside getters and setters.
47    public void setAge(int age) {
48        if(age>=16) {
49            this.age=age;
50            System.out.println("hello olders");
51        }
52        else {
53            System.out.println("grow up to 16 years");
54        }
55    }
56
57    public void setAdhar(String adhar) {
58        this.adhar = adhar;
59    }
60
61    public void setPhone(String phone) {
62        this.phone = phone;
63    }
64}
65
```

paste bank4

```

bank4.java X
1 package com.day16;
2
3 public class bank4 {
4
5     private final String name="gorilla";
6     private int age;
7     private String adhar;
8     private String phone;
9
10    // Constructor
11@   public bank4(int age, String adhar, String phone) {
12        this.age = age;
13
14        //same setter condition can be set in constructor also.
15        //sysout doesnt come in constructor.
16        //constructor only for initialising and setting values not returning anything.
17        //because constructor doesnt have return type.
18@       if(age>=16) {
19            this.age=age;
20            sysout.println("hello olders");
21        }
22@       else {
23            sysout.println("grow up to 16 years");
24        }
25        this.adhar = adhar;
26        this.phone = phone;
27    }
28
29@   public bank4() {
30        // TODO Auto-generated constructor stub
31    }
32
33    // Getter
34@   public String getName() {
35        return name;
36    }
37
38@   public int getAge() {
39        return age;
40    }
41
42@   public String getAdhar() {
43        return adhar;
44    }
45
46@   public String getPhone() {
47        return phone;
48    }
49
50    // Setter
51@   public void setName(String name) {
52        this.name = name; //The final field bank2.name cannot be assigned
53    }

```

```
51  public void setName(String name) {
52      this.name = name; //The final field bank2.name cannot be assigned
53  }
54
55  //we can write any damn code inside getters and setters.
56  public void setAge(int age) {
57      if(age>=16) {
58          this.age=age;
59          System.out.println("hello olders");
60      }
61      else {
62          System.out.println("grow up to 16 years");
63      }
64  }
65
66  public void setAdhar(String adhar) {
67      this.adhar = adhar;
68  }
69
70  public void setPhone(String phone) {
71      this.phone = phone;
72  }
73 }
74 }
```

paste bank 5

The screenshot shows a Java code editor window with the file 'bank5.java' open. The code defines a class 'bank5' with private fields for name, age, adhar, and phone. It includes a constructor that initializes age based on user input, and two getter methods for name and age. A TODO comment is present in the constructor stub.

```
1 package com.day16;
2
3 public class bank5 {
4
5     private final String name="gorilla";
6     private int age;
7     private String adhar;
8     private String phone;
9
10    // Constructor
11    public bank5(int age, String adhar, String phone) {
12        this.age = age;
13
14        if(age>=16) {
15            this.age=age;
16        }
17        else {
18            this.age=10;
19        }
20        this.adhar = adhar;
21        this.phone = phone;
22    }
23
24    public bank5() {
25        // TODO Auto-generated constructor stub
26    }
27
28    // Getter
29    public String getName() {
30        return name;
31    }
32
33    public int getAge() {
34        return age;
35    }
36}
```

```
34         return age;
35     }
36
37     public String getAdhar() {
38         return adhar;
39     }
40
41     public String getPhone() {
42         return phone;
43     }
44
45     // Setter
46     public void setName(String name) {
47         this.name = name; //The final field bank2.name cannot be assigned
48     }
49
50     //we can write any damn code inside getters and setters.
51     public void setAge(int age) {
52         if(age>=16) {
53             this.age=age;
54             System.out.println("hello olders");
55         }
56         else {
57             System.out.println("grow up to 16 years");
58         }
59     }
60
61     public void setAdhar(String adhar) {
62         this.adhar = adhar;
63     }
64
65     public void setPhone(String phone) {
66         this.phone = phone;
67     }
68 }
69 }
```

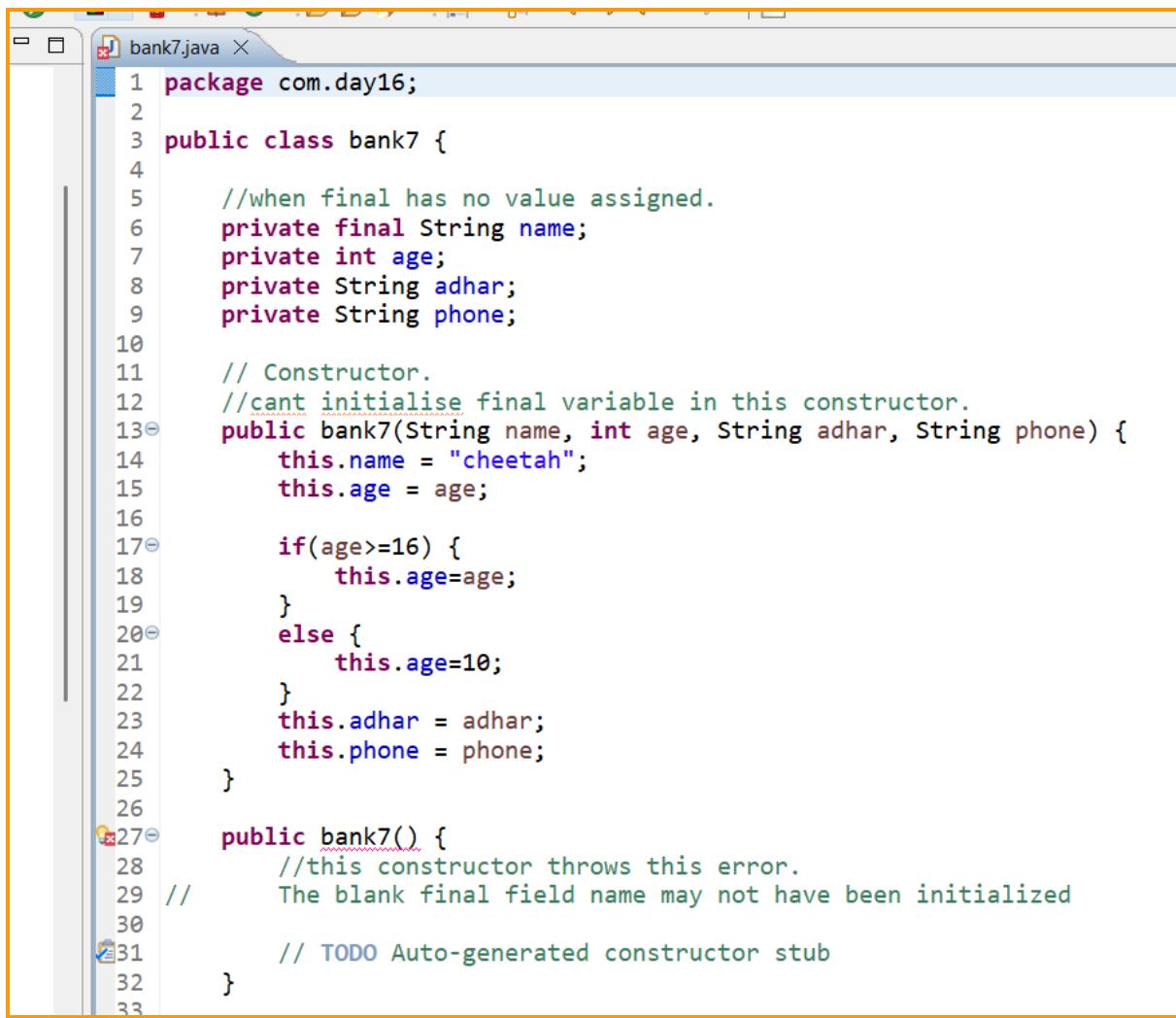
paste bank6

The screenshot shows a Java code editor window with the file 'bank6.java' open. The code defines a class 'bank6' with private fields for name, age, adhar, and phone. It includes a constructor that initializes age based on a condition, and two getter methods for name and age. The code is annotated with line numbers and some comments.

```
1 package com.day16;
2
3 public class bank6 {
4
5     //when final has value assigned.
6     private final String name="gorilla";
7     private int age;
8     private String adhar;
9     private String phone;
10
11    // Constructor
12    public bank6(int age, String adhar, String phone) {
13        this.age = age;
14
15        if(age>=16) {
16            this.age=age;
17        }
18        else {
19            this.age=10;
20        }
21        this.adhar = adhar;
22        this.phone = phone;
23    }
24
25    public bank6() {
26        // TODO Auto-generated constructor stub
27    }
28
29    // Getter
30    public String getName() {
31        return name;
32    }
33
34    public int getAge() {
35        return age;
36    }
37}
```

```
34    public int getAge() {
35        return age;
36    }
37
38    public String getAdhar() {
39        return adhar;
40    }
41
42    public String getPhone() {
43        return phone;
44    }
45
46    //we can write any damn code inside getters and setters.
47    public void setAge(int age) {
48        if(age>=16) {
49            this.age=age;
50            System.out.println("hello olders");
51        }
52        else {
53            System.out.println("grow up to 16 years");
54        }
55    }
56
57    public void setAdhar(String adhar) {
58        this.adhar = adhar;
59    }
60
61    public void setPhone(String phone) {
62        this.phone = phone;
63    }
64
65    public static void main(String[] args) {
66
67        bank6 obj = new bank6(20, "adhar", "phone");
68        System.out.println(obj.getName());
69        System.out.println(obj.getAge());
70        System.out.println(obj.getAdhar());
71        System.out.println(obj.getPhone());
72    }
73
74
75 //gorilla
76 //20
77 //adhar
78 //phone
79
80
```

## paste bank7



The screenshot shows a Java code editor window with the file "bank7.java" open. The code defines a class named "bank7" with private final fields for name, age, adhar, and phone. It includes a constructor that initializes the name to "cheetah" and sets age based on a condition. There is also a blank constructor stub. The code is annotated with comments explaining the usage of final variables.

```
1 package com.day16;
2
3 public class bank7 {
4
5     //when final has no value assigned.
6     private final String name;
7     private int age;
8     private String adhar;
9     private String phone;
10
11    // Constructor.
12    //cant initialise final variable in this constructor.
13    public bank7(String name, int age, String adhar, String phone) {
14        this.name = "cheetah";
15        this.age = age;
16
17        if(age>=16) {
18            this.age=age;
19        }
20        else {
21            this.age=10;
22        }
23        this.adhar = adhar;
24        this.phone = phone;
25    }
26
27    public bank7() {
28        //this constructor throws this error.
29        // The blank final field name may not have been initialized
30
31        // TODO Auto-generated constructor stub
32    }
33}
```

```
31         // TODO Auto-generated constructor stub
32     }
33
34     // Getter
35     public String getName() {
36         return name;
37     }
38
39     public int getAge() {
40         return age;
41     }
42
43     public String getAdhar() {
44         return adhar;
45     }
46
47     public String getPhone() {
48         return phone;
49     }
50
51     //we can write any damn code inside getters and setters.
52     public void setAge(int age) {
53         if(age>=16) {
54             this.age=age;
55             System.out.println("hello olders");
56         }
57         else {
58             System.out.println("grow up to 16 years");
59         }
60     }
61
62     public void setAdhar(String adhar) {
63         this.adhar = adhar;
64     }
65
66     public void setPhone(String phone) {
67         this.phone = phone;
68     }
69
70     public static void main(String[] args) {
71
72         bank7 obj = new bank7("manking", 20, "adhar", "phone");
73         System.out.println(obj.getName());
74         System.out.println(obj.getAge());
75         System.out.println(obj.getAdhar());
76         System.out.println(obj.getPhone());
77     }
78 }
```

paste bank8

```
1 package com.day16;
2
3 public class bank8 {
4
5     //when final has no value assigned.
6     private final String name;
7     private int age;
8     private String adhar;
9     private String phone;
10
11    // Constructor.
12
13    public bank8(String name, int age, String adhar, String phone) {
14        this.name = "stone cold";
15        this.age = age;
16
17        if(age>=16) {
18            this.age=age;
19        }
20        else {
21            this.age=10;
22        }
23        this.adhar = adhar;
24        this.phone = phone;
25    }
26
27    // Getter
28    public String getName() {
29        return name;
30    }
31
32    public int getAge() {
33        return age;
34    }
35}
```

```

33         return age;
34     }
35
36     public String getAdhar() {
37         return adhar;
38     }
39
40     public String getPhone() {
41         return phone;
42     }
43
44     //we can write any damn code inside getters and setters.
45     public void setAge(int age) {
46         if(age>=16) {
47             this.age=age;
48             System.out.println("hello olders");
49         }
50         else {
51             System.out.println("grow up to 16 years");
52         }
53     }
54
55     public void setAdhar(String adhar) {
56         this.adhar = adhar;
57     }
58
59     public void setPhone(String phone) {
60         this.phone = phone;
61     }
62
63
64
65     public static void main(String[] args) {
66
67         bank8 obj = new bank8(getna, "adhar", "phone"); //we cannot use getname method directly.
68         //first we need to create the object then only can access.
69         System.out.println(obj.getName());
70         System.out.println(obj.getAge());
71         System.out.println(obj.getAdhar());
72         System.out.println(obj.getPhone());
73     }
74
75     //gorilla
76     //20
77     //adhar
78     //phone
79

```

paste bank9



```

36         return adhar;
37     }
38
39     public String getPhone() {
40         return phone;
41     }
42
43     //we can write any damn code inside getters and setters.
44     public void setAge(int age) {
45         if(age>=16) {
46             this.age=age;
47             System.out.println("hello olders");
48         }
49         else {
50             System.out.println("grow up to 16 years");
51         }
52     }
53
54     public void setAdhar(String adhar) {
55         this.adhar = adhar;
56     }
57
58     public void setPhone(String phone) {
59         this.phone = phone;
60     }
61
62     public static void main(String[] args) {
63
64         public static void main(String[] args) {
65
66             //allows to change value of final but doesn't reflect in output.
67             //in output can see original value only.
68             bank9 obj = new bank9("manking", 20, "adhar", "phone");
69             System.out.println(obj.getName());
70             System.out.println(obj.getAge());
71             System.out.println(obj.getAdhar());
72             System.out.println(obj.getPhone());
73         }
74     }
75
76     //cheetah
77     //20
78     //adhar
79     //phone

```

paste bank10

```
1 package com.day16;
2
3 public class bank10 {
4
5     //when final has no value assigned.
6     private final String name;
7     private int age;
8     private String adhar;
9     private String phone;
10
11    // Constructor.
12    //now we get error in this constructor.
13    //so final has to be initialised in every constructor if there are more than one
14    //constructor in class.
15    public bank10(int age, String adhar, String phone) {
16
17        this.age = age;
18
19        if(age>=16) {
20            this.age=age;
21        }
22        else {
23            this.age=10;
24        }
25        this.adhar = adhar;
26        this.phone = phone;
27    }
28
29    //create default constructor to initialise final variable.
30    public bank10() {
31        this.name = "gorilla";
32    }
33
34    // Getter
35    public String getName() {
36        return name;
37    }
38}
```

```
36         return name;
37     }
38
39 $\Theta$      public int getAge() {
40         return age;
41     }
42
43 $\Theta$      public String getAdhar() {
44         return adhar;
45     }
46
47 $\Theta$      public String getPhone() {
48         return phone;
49     }
50
51     //we can write any damn code inside getters and setters.
52 $\Theta$      public void setAge(int age) {
53         if(age>=16) {
54             this.age=age;
55             System.out.println("hello olders");
56         }
57         else {
58             System.out.println("grow up to 16 years");
59         }
60     }
61
62 $\Theta$      public void setAdhar(String adhar) {
63         this.adhar = adhar;
64     }
```

```
63         this.adhar = adhar;
64     }
65
66    public void setPhone(String phone) {
67        this.phone = phone;
68    }
69
70    public static void main(String[] args) {
71
72        bank10 obj = new bank10(10, "adhar", "phone");
73        System.out.println(obj.getName());
74        System.out.println(obj.getAge());
75        System.out.println(obj.getAdhar());
76        System.out.println(obj.getPhone());
77    }
78 }
79
80 //gorilla
81 //20
82 //adhar
83 //phone
84
85
```

paste bank11

```
1 package com.day16;
2
3 public class bank11 {
4
5     //when final has no value assigned.
6     private final String name;
7     private int age;
8     private String adhar;
9     private String phone;
10
11    // Constructor.
12    //initialise final variable blank here.
13    public bank11(int age, String adhar, String phone) {
14
15        this.name = "";
16        this.age = age;
17
18        if(age>=16) {
19            this.age=age;
20        }
21        else {
22            this.age=10;
23        }
24        this.adhar = adhar;
25        this.phone = phone;
26    }
27
28    //create default constructor to initialise final variable.
29    public bank11() {
30        this.name = "gorilla";
31    }
32
33    // Getter
34    public String getName() {
35        return name;
36    }
37}
```

```
35     return name;
36 }
37
38 public int getAge() {
39     return age;
40 }
41
42 public String getAdhar() {
43     return adhar;
44 }
45
46 public String getPhone() {
47     return phone;
48 }
49
50 //we can write any damn code inside getters and setters.
51 public void setAge(int age) {
52     if(age>=16) {
53         this.age=age;
54         System.out.println("hello olders");
55     }
56     else {
57         System.out.println("grow up to 16 years");
58     }
59 }
60
61 public void setAdhar(String adhar) {
62     this.adhar = adhar;
63 }
64
65 public void setPhone(String phone) {
66     this.phone = phone;
67 }
```

```
66         this.phone = phone;
67     }
68
69     public static void main(String[] args) {
70
71         //create object of empty constructor.
72         bank11 obj1 = new bank11();
73         System.out.println(obj1.getName());
74
75         bank11 obj = new bank11(10, "adhar", "phone");
76         System.out.println(obj.getName()); //blank
77         System.out.println(obj.getAge());
78         System.out.println(obj.getAdhar());
79         System.out.println(obj.getPhone());
80     }
81 }
82
83 //gorilla
84 //
85 //10
86 //adhar
87 //phone
88
89
```

paste bank12

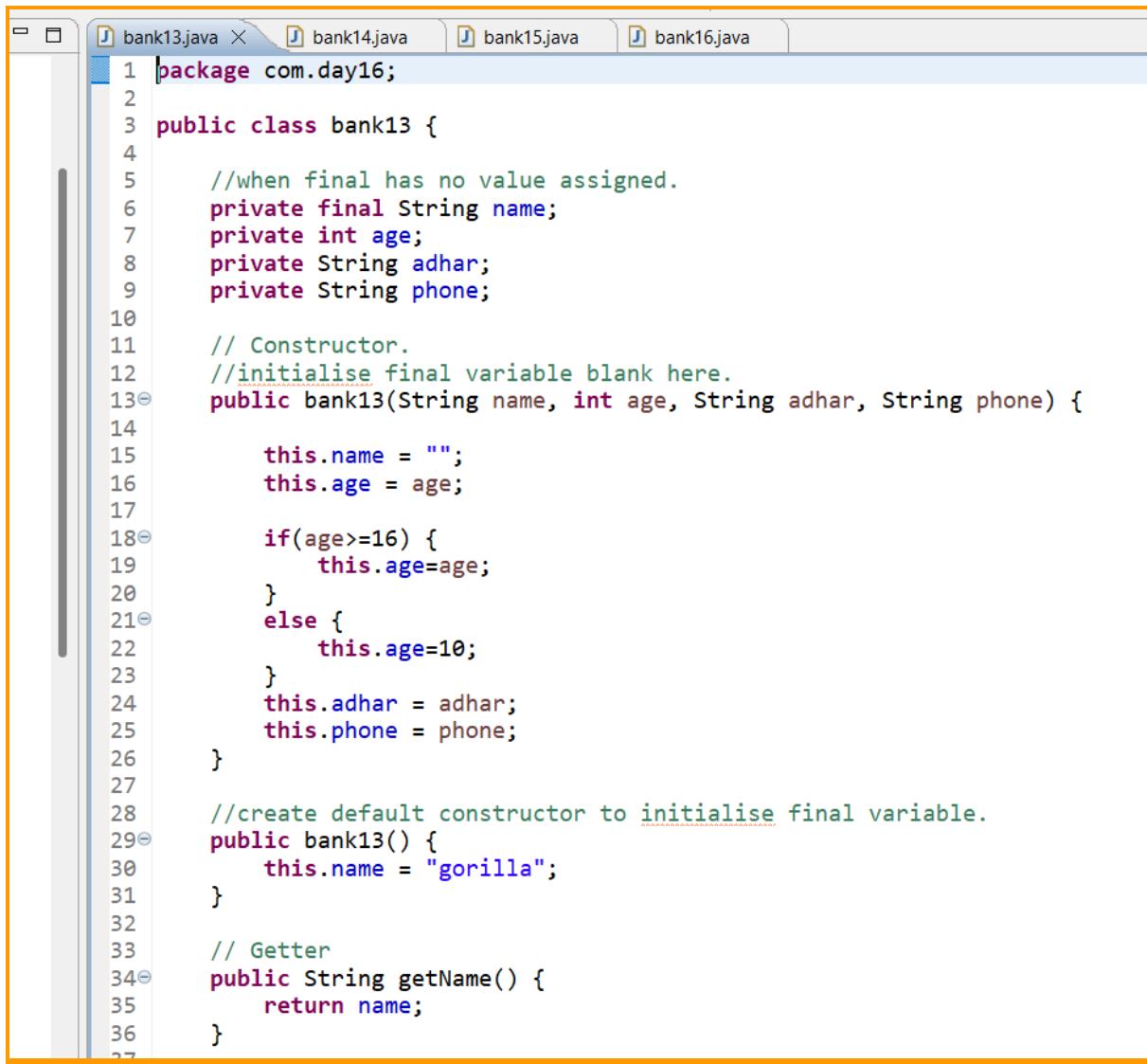
The screenshot shows a Java code editor window with the file `bank12.java` open. The code defines a class `bank12` with private final fields `name`, `age`, `adhar`, and `phone`. It includes a constructor that initializes these fields based on the age provided, with a default value of 10 if age is less than or equal to 16. A default constructor sets the name to "gorilla". A getter method `getName()` is also present. The code is annotated with comments explaining the logic for initializing final variables and creating a default constructor.

```
1 package com.day16;
2
3 public class bank12 {
4
5     //when final has no value assigned.
6     private final String name;
7     private int age;
8     private String adhar;
9     private String phone;
10
11    // Constructor.
12    //initialise final variable blank here.
13    public bank12(String name, int age, String adhar, String phone) {
14
15        this.name = "";
16        this.age = age;
17
18        if(age>=16) {
19            this.age=age;
20        }
21        else {
22            this.age=10;
23        }
24        this.adhar = adhar;
25        this.phone = phone;
26    }
27
28    //create default constructor to initialise final variable.
29    public bank12() {
30        this.name = "gorilla";
31    }
32
33    // Getter
34    public String getName() {
35        return name;
36    }
37}
```

```
35         return name;
36     }
37
38⊕    public int getAge() {
39        return age;
40    }
41
42⊕    public String getAdhar() {
43        return adhar;
44    }
45
46⊕    public String getPhone() {
47        return phone;
48    }
49
50    //we can write any damn code inside getters and setters.
51⊕    public void setAge(int age) {
52⊕        if(age>=16) {
53            this.age=age;
54            System.out.println("hello olders");
55        }
56⊕        else {
57            System.out.println("grow up to 16 years");
58        }
59    }
60
61⊕    public void setAdhar(String adhar) {
62        this.adhar = adhar;
63    }
```

```
62     public void setAdhar(String adhar) {
63         this.adhar = adhar;
64     }
65     public void setPhone(String phone) {
66         this.phone = phone;
67     }
68
69     public static void main(String[] args) {
70
71         //create object of empty constructor.
72         bank12 obj1 = new bank12();
73         System.out.println(obj1.getName());
74
75         //still we get blank for this final here.
76         bank12 obj = new bank12(obj1.getName(),10, "adhar", "phone");
77         System.out.println(obj.getName()); //blank
78         System.out.println(obj.getAge());
79         System.out.println(obj.getAdhar());
80         System.out.println(obj.getPhone());
81     }
82 }
83
84 //gorilla
85 //
86 //10
87 //adhar
88 //phone
89
90
91
```

paste bank13

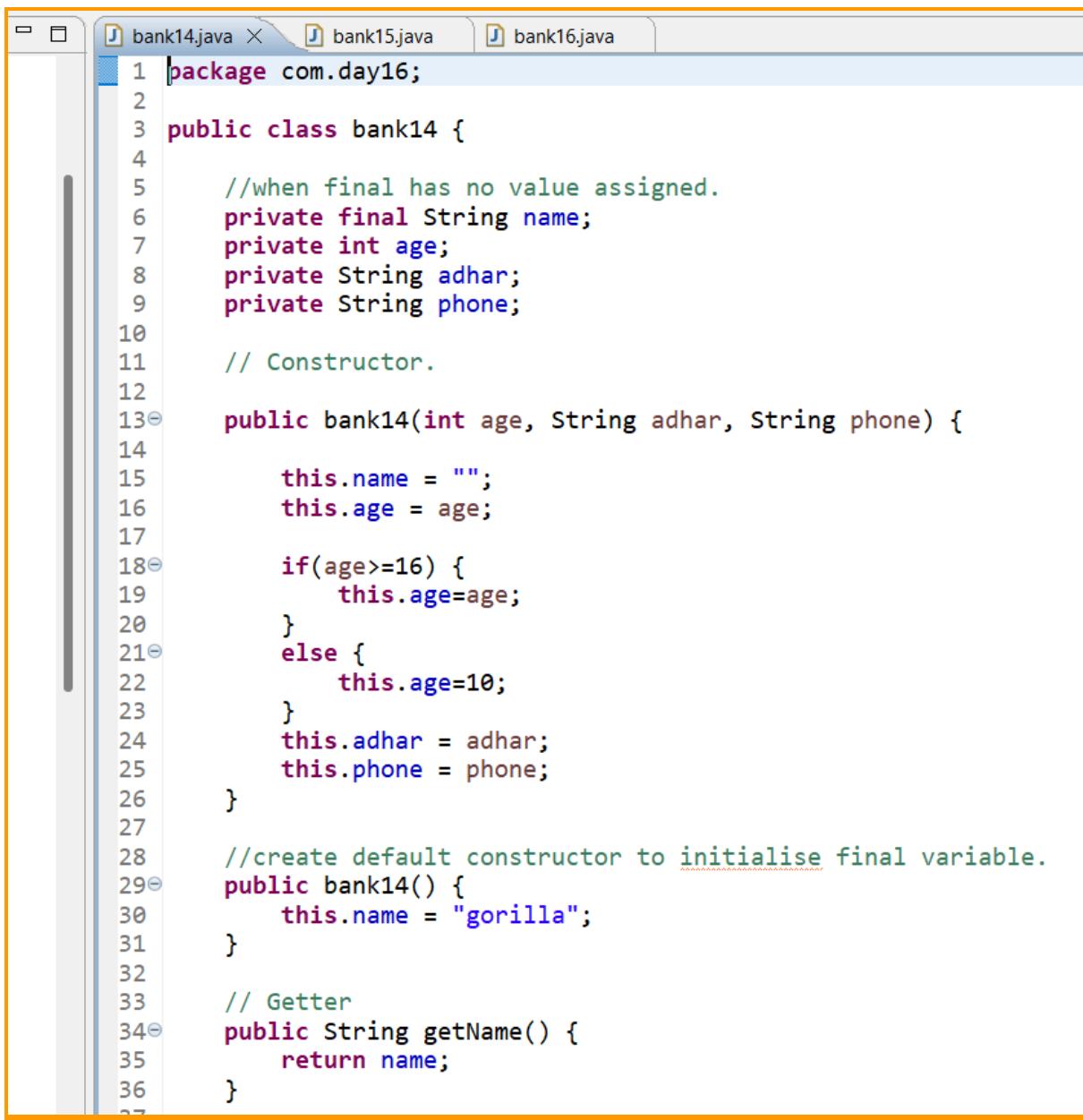


```
1 package com.day16;
2
3 public class bank13 {
4
5     //when final has no value assigned.
6     private final String name;
7     private int age;
8     private String adhar;
9     private String phone;
10
11    // Constructor.
12    //initialise final variable blank here.
13    public bank13(String name, int age, String adhar, String phone) {
14
15        this.name = "";
16        this.age = age;
17
18        if(age>=16) {
19            this.age=age;
20        }
21        else {
22            this.age=10;
23        }
24        this.adhar = adhar;
25        this.phone = phone;
26    }
27
28    //create default constructor to initialise final variable.
29    public bank13() {
30        this.name = "gorilla";
31    }
32
33    // Getter
34    public String getName() {
35        return name;
36    }
37}
```

```
34     public String getName() {
35         return name;
36     }
37
38     public int getAge() {
39         return age;
40     }
41
42     public String getAdhar() {
43         return adhar;
44     }
45
46     public String getPhone() {
47         return phone;
48     }
49
50     //we can write any damn code inside getters and setters.
51     public void setAge(int age) {
52         if(age>=16) {
53             this.age=age;
54             System.out.println("hello olders");
55         }
56         else {
57             System.out.println("grow up to 16 years");
58         }
59     }
60 }
```

```
57         System.out.println("grow up to 16 years");
58     }
59 }
60
61 public void setAdhar(String adhar) {
62     this.adhar = adhar;
63 }
64
65 public void setPhone(String phone) {
66     this.phone = phone;
67 }
68
69 public static void main(String[] args) {
70
71     //create object of empty constructor.
72     bank13 obj1 = new bank13();
73     System.out.println(obj1.getName());
74     String s1=obj1.getName();
75
76     //still we get blank as this constructor is initialising it to blank.
77     bank13 obj = new bank13(s1,10, "adhar", "phone");
78     System.out.println(obj.getName()); //blank
79     System.out.println(obj.getAge());
80     System.out.println(obj.getAdhar());
81     System.out.println(obj.getPhone());
82 }
83 }
84
85 //gorilla
86 //
87 //10
88 //adhar
89 //phone
90
91
92
93
```

paste bank14

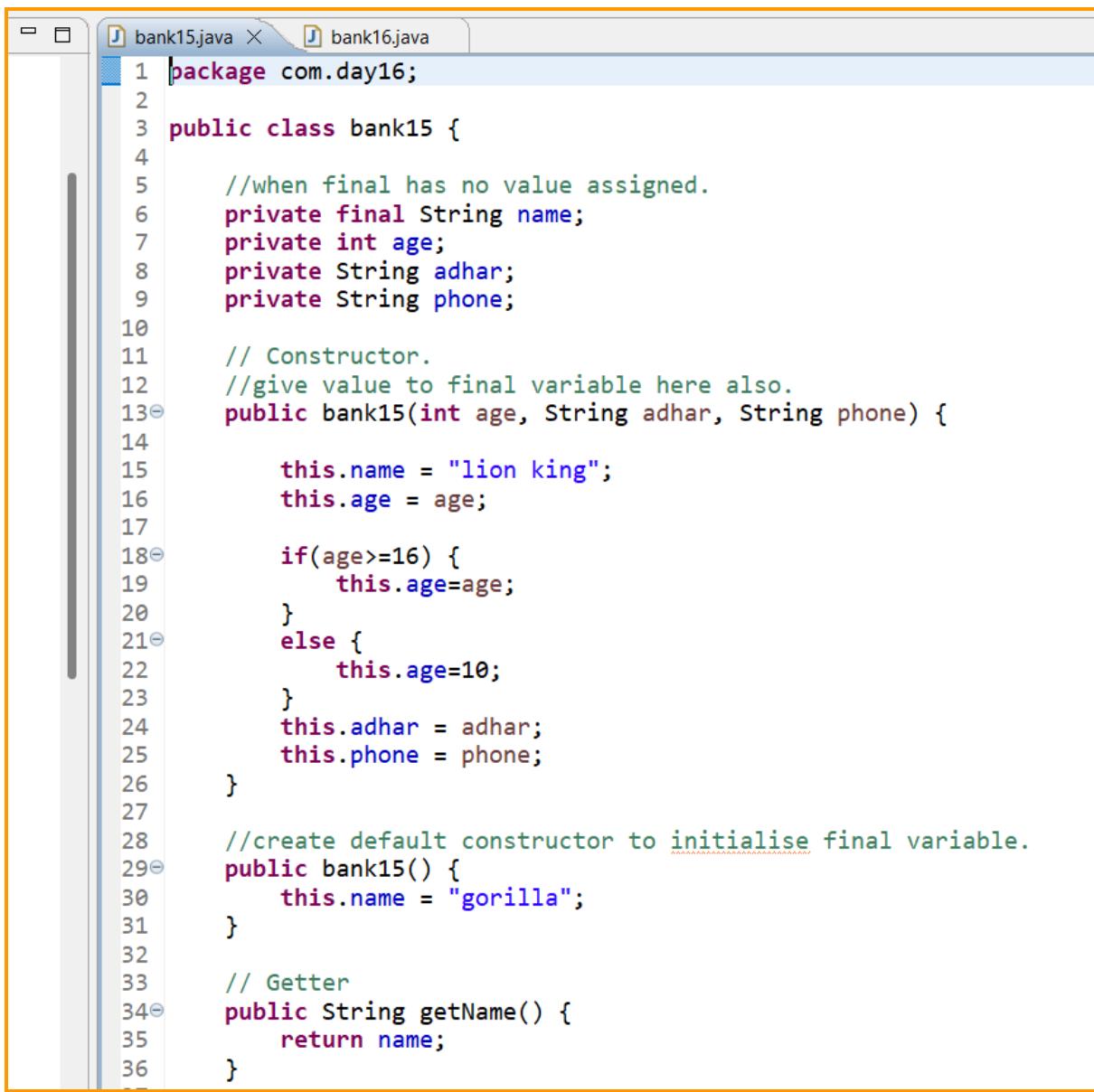


```
1 package com.day16;
2
3 public class bank14 {
4
5     //when final has no value assigned.
6     private final String name;
7     private int age;
8     private String adhar;
9     private String phone;
10
11    // Constructor.
12
13    public bank14(int age, String adhar, String phone) {
14
15        this.name = "";
16        this.age = age;
17
18        if(age>=16) {
19            this.age=age;
20        }
21        else {
22            this.age=10;
23        }
24        this.adhar = adhar;
25        this.phone = phone;
26    }
27
28    //create default constructor to initialise final variable.
29    public bank14() {
30        this.name = "gorilla";
31    }
32
33    // Getter
34    public String getName() {
35        return name;
36    }
37}
```

```
35         return name;
36     }
37
38     public int getAge() {
39         return age;
40     }
41
42     public String getAdhar() {
43         return adhar;
44     }
45
46     public String getPhone() {
47         return phone;
48     }
49
50     //we can write any damn code inside getters and setters.
51     public void setAge(int age) {
52         if(age>=16) {
53             this.age=age;
54             System.out.println("hello olders");
55         }
56         else {
57             System.out.println("grow up to 16 years");
58         }
59     }
60 }
```

```
57          System.out.println("grow up to to y");
58      }
59  }
60
61@ public void setAdhar(String adhar) {
62     this.adhar = adhar;
63 }
64
65@ public void setPhone(String phone) {
66     this.phone = phone;
67 }
68
69@ public static void main(String[] args) {
70
71     //create object of empty constructor.
72     bank14 obj1 = new bank14();
73     System.out.println(obj1.getName());
74     String s1=obj1.getName();
75     System.out.println(s1);
76
77     System.out.println(obj1.getName());
78     String s1=obj1.getName();
79     System.out.println(s1);
80
81     //still we get blank as this constructor is initialising it to blank.
82     bank14 obj = new bank14(10, "adhar", "phone");
83     System.out.println(obj.getName()); //blank
84     System.out.println(obj.getAge());
85     System.out.println(obj.getAdhar());
86     System.out.println(obj.getPhone());
87
88
89 //10
90 //adhar
91 //phone
92
93
94
95
96
```

paste bank15



```
1 package com.day16;
2
3 public class bank15 {
4
5     //when final has no value assigned.
6     private final String name;
7     private int age;
8     private String adhar;
9     private String phone;
10
11    // Constructor.
12    //give value to final variable here also.
13    public bank15(int age, String adhar, String phone) {
14
15        this.name = "lion king";
16        this.age = age;
17
18        if(age>=16) {
19            this.age=age;
20        }
21        else {
22            this.age=10;
23        }
24        this.adhar = adhar;
25        this.phone = phone;
26    }
27
28    //create default constructor to initialise final variable.
29    public bank15() {
30        this.name = "gorilla";
31    }
32
33    // Getter
34    public String getName() {
35        return name;
36    }
}
```

```
35     public String getName() {
36         return name;
37     }
38     public int getAge() {
39         return age;
40     }
41
42     public String getAdhar() {
43         return adhar;
44     }
45
46     public String getPhone() {
47         return phone;
48     }
49
50     //we can write any damn code inside getters and setters.
51     public void setAge(int age) {
52         if(age>=16) {
53             this.age=age;
54             System.out.println("hello olders");
55         }
56         else {
57             System.out.println("grow up to 16 years");
58         }
59     }
60
61     public void setAdhar(String adhar) {
62         this.adhar = adhar;
63     }
64
```

```
62         this.adhar = adhar;
63     }
64
65     public void setPhone(String phone) {
66         this.phone = phone;
67     }
68
69     public static void main(String[] args) {
70
71         //create object of empty constructor.
72         bank15 obj1 = new bank15();
73         System.out.println(obj1.getName());
74         String s1=obj1.getName();
75         System.out.println(s1);
76
77
78         bank15 obj = new bank15(10, "adhar", "phone");
79         System.out.println(obj.getName());
80         System.out.println(obj.getAge());
81         System.out.println(obj.getAdhar());
82         System.out.println(obj.getPhone());
83     }
84 }
85
86 //gorilla
87 //gorilla
88 //lion king
89 //10
90 //adhar
91 //phone
92
93
94
95
96
97
```

paste bank16

The screenshot shows a Java code editor window with the file 'bank16.java' open. The code defines a class 'bank16' with private final variables for name, age, adhar, and phone. It includes a constructor that initializes these variables and handles age validation. A default constructor is also provided to initialize the final variables.

```
1 package com.day16;
2
3 public class bank16 {
4
5     //when final has no value assigned.
6     private final String name;
7     private int age;
8     private String adhar;
9     private String phone;
10
11    // Constructor.
12    //give value to final variable here also.
13    //give as parameter also.
14    public bank16(String name, int age, String adhar, String phone) {
15
16        this.name = "lion king";
17        this.age = age;
18
19        if(age>=16) {
20            this.age=age;
21        }
22        else {
23            this.age=10;
24        }
25        this.adhar = adhar;
26        this.phone = phone;
27    }
28
29    //create default constructor to initialise final variable.
30    public bank16() {
31        this.name = "gorilla";
32    }
33}
```

```
31         this.name = "gorilla";
32     }
33
34     // Getter
35     public String getName() {
36         return name;
37     }
38
39     public int getAge() {
40         return age;
41     }
42
43     public String getAdhar() {
44         return adhar;
45     }
46
47     public String getPhone() {
48         return phone;
49     }
50
51     //we can write any damn code inside getters and setters.
52     public void setAge(int age) {
53         if(age>=16) {
54             this.age=age;
55             System.out.println("hello olders");
56         }
57         else {
58             System.out.println("grow up to 16 years");
59         }
60     }
61 }
```

```

58         System.out.println( " grow up to 10 years " );
59     }
60 }
61
62 public void setAdhar(String adhar) {
63     this.adhar = adhar;
64 }
65
66 public void setPhone(String phone) {
67     this.phone = phone;
68 }
69
70 public static void main(String[] args) {
71
72     //create object of empty constructor.
73     bank16 obj1 = new bank16();
74     System.out.println(obj1.getName());
75     String s1=obj1.getName();
76     System.out.println(s1);
77
78     //cant change final value from object creation step.
79     bank16 obj = new bank16("vince macmahon", 10, "adhar", "phone");
80     System.out.println(obj.getName());
81     System.out.println(obj.getAge());
82     System.out.println(obj.getAdhar());
83     System.out.println(obj.getPhone());
84 }
85 }
86
87 //gorilla
88 //gorilla
89 //lion king
90 //10
91 //adhar
92 //phone
93
94
95
96
97
98

```

Same condition can be set in constructor also-

```
9
10 public Bank(String name, int age, String aadharNumber, String phone) {
11     this.name = name;
12
13     if (age >= 15) {
14         this.age = age;
15     }
16
17     this.aadharNumber = aadharNumber;
18     this.phone = phone;
19 }
20
```



A screenshot of a Java Integrated Development Environment (IDE). The top menu bar shows various project files like Employee.java, TestEmp.java, Browser.java, LoginPage.java, Bank.java, and TestBank.java. The main editor area displays the following Java code:

```
1 package EncapsulationConcept;
2
3 public class TestBank {
4
5     public static void main(String[] args) {
6
7         Bank b = new Bank("Pooja", 20, "21212121", "21212121");
8
9         System.out.println(b.getAadharNumber());
10
11
12
13
14     }
15
16 }
17
```

The console tab in the IDE shows the output of the program:

```
Console [ ] Prok
<terminated> TestBank [J
21212121
```

The screenshot shows the Eclipse IDE interface. The top part displays the code for `TestBank.java` within the `EncapsulationConcept` package. The code creates a `Bank` object named `b` with the name "Pooja", age 5, Aadhar number "21212121", and PAN number "21212121". It then prints the Aadhar number and age using `System.out.println`. The bottom part shows the `Console` view with the output: "21212121" followed by "0".

```
1 package EncapsulationConcept;
2
3 public class TestBank {
4
5     public static void main(String[] args) {
6
7         Bank b = new Bank("Pooja", 5, "21212121", "21212121");
8
9         System.out.println(b.getAadharNumber());
10
11        System.out.println(b.getAge());
12
13
14    }
15
16
17 }
18
```

Console

```
<terminated> TestBank [J
21212121
0
```

paste bank17 and test bank17

```
bank17.java X
1 package com.day16;
2
3 public class bank17 {
4
5     private String name;
6     private int age;
7     private String adhar;
8     private String phone;
9
10    // Constructor
11    public bank17(String name, int age, String adhar, String phone) {
12
13        this.name=name;
14
15        this.age = age;
16
17        //same setter condition can be set in constructor also.
18        if(age>=16) {
19            this.age=age;
20        }
21        else {
22            this.age=10;
23        }
24        this.adhar = adhar;
25        this.phone = phone;
26    }
27
28    public bank17() {
29        // TODO Auto-generated constructor stub
30    }
31
32    // Getter
33    public String getName() {
34        return name;
35    }
36
```

```
34     return name;
35 }
36
37 @
38     public int getAge() {
39         return age;
40     }
41 @
42     public String getAdhar() {
43         return adhar;
44     }
45 @
46     public String getPhone() {
47         return phone;
48     }
49     // Setter
50 @
51     public void setName(String name) {
52         this.name = name;
53     }
54     //we can write any damn code inside getters and setters.
55 //     public void setAge(int age) {
56 //         if(age>=16) {
57 //             this.age=age;
58 //             System.out.println("hello olders");
59 //         }
60 //         else {
61 //             System.out.println("grow up to 16 years");
62 //         }
63 //     }
64 }
```

```

62 //      }
63 //    }
64
65 public void setAdhar(String adhar) {
66     this.adhar = adhar;
67 }
68
69 public void setPhone(String phone) {
70     this.phone = phone;
71 }
72 }
73

```

testbank17.java

```

1 package com.day16;
2
3 public class testbank17 {
4
5     public static void main(String[] args) {
6
7         bank17 obj = new bank17("manking", 20, "adhar", "phone");
8         System.out.println(obj.getName());
9         System.out.println(obj.getAge());
10        System.out.println(obj.getAdhar());
11        System.out.println(obj.getPhone());
12
13        //create another object with different values.
14
15        bank17 obj1 = new bank17("vince macmahon", 10, "adhar", "phone");
16        System.out.println(obj1.getName());
17        System.out.println(obj1.getAge());
18        System.out.println(obj1.getAdhar());
19        System.out.println(obj1.getPhone());
20    }
21
22 }
23
24
25 //manking
26 //20
27 //adhar
28 //phone
29
30 //vince macmahon
31 //10
32 //adhar
33 //phone
34

```

Actual business logic looks like this-

See the difference between the ones written in setter and constructor and here.

```

50
57     //business logic method:
58     public void openAccount() {
59         if(this.age>=15) {
60             System.out.println("opening the account");
61             //...
62             //..
63         }
64         else {
65             System.out.println("age is less than 15");
66         }
67     }
68
69 }
```

```

1 package EncapsulationConcept;
2
3 public class TestBank {
4
5     public static void main(String[] args) {
6
7         Bank b = new Bank("Pooja", 5, "21212121", "21212121");
8
9         System.out.println(b.getAadharNumber());
10
11        System.out.println(b.getAge());
12
13        b.openAccount();
14
15    }
16 }
```

```

Console Problems Javadoc Decl
<terminated> TestBank [Java Application] /Users/navee
21212121
0
age is less than 15
```

Valid age-

```

5     public static void main(String[] args) {
6
7         Bank b = new Bank("Pooja", 50, "21212121", "21212121");
8 }
```

The screenshot shows the Eclipse IDE interface. The Java Editor window contains the following code:

```

1 package EncapsulationConcept;
2
3 public class Compnay {
4
5
6     private Compnay() {
7
8 }
9
10
11
12
13 }
14

```

The 'Compnay' class has a private constructor. The Java Editor shows line numbers 1 through 14. The 'Compnay()' constructor is highlighted in blue.

The Console view at the bottom displays the following output:

```

Console Problems Javadoc Declaration TestNG
<terminated> TestBank [Java Application] /Users/naveenautomationlabs/.p2/pool/plugins/org.eclipse.jdt.core_3.12.0.v20150514-1200.jar
21212121
50
opening the account, AC is ready to use

```

Private constructor-

Allowed.

But we cannot create object of class.

The screenshot shows the Eclipse IDE interface. The Java Editor window contains the same code as before, but the 'Compnay()' constructor is now underlined in red, indicating a syntax error.

The Problems view at the top shows one error:

```

Console Problems Javadoc Declaration TestNG
<terminated> TestBank [Java Application] /Users/naveenautomationlabs/.p2/pool/plugins/org.eclipse.jdt.core_3.12.0.v20150514-1200.jar
21212121
50
opening the account, AC is ready to use

```

The error message is:

```

private Compnay() {
^
1 error found

```

Error.

```

15
16     Compnay co = new Compnay();
17
18 }

```

//The constructor privateconst2() is not visible  
when we try to access from another class.

**paste privateconst1**

```

1 package com.day16;
2
3 public class privateconst1 {
4
5     private privateconst1() {
6         System.out.println("this is private constructor.");
7     }
8
9     //in same class private works.
10    public static void main(String[] args) {
11        privateconst1 obj = new privateconst1();
12    }
13 }
14
15 //this is private constructor.
16

```

We have methods but constructor is private-

```

1 package EncapsulationConcept;
2
3 public class Compnay {
4
5
6     private Compnay() {
7
8     }
9
10
11    public static void getRevenue() {
12        System.out.println("get rev");
13    }
14
15
16 }
17
18
19     Compnay.getRevenue();
20
21

```

We create the method as static so can be accessed with class name.

Constructor for dealing with objects not class names.

```
get rev I
```

paste company1 and testcompany1

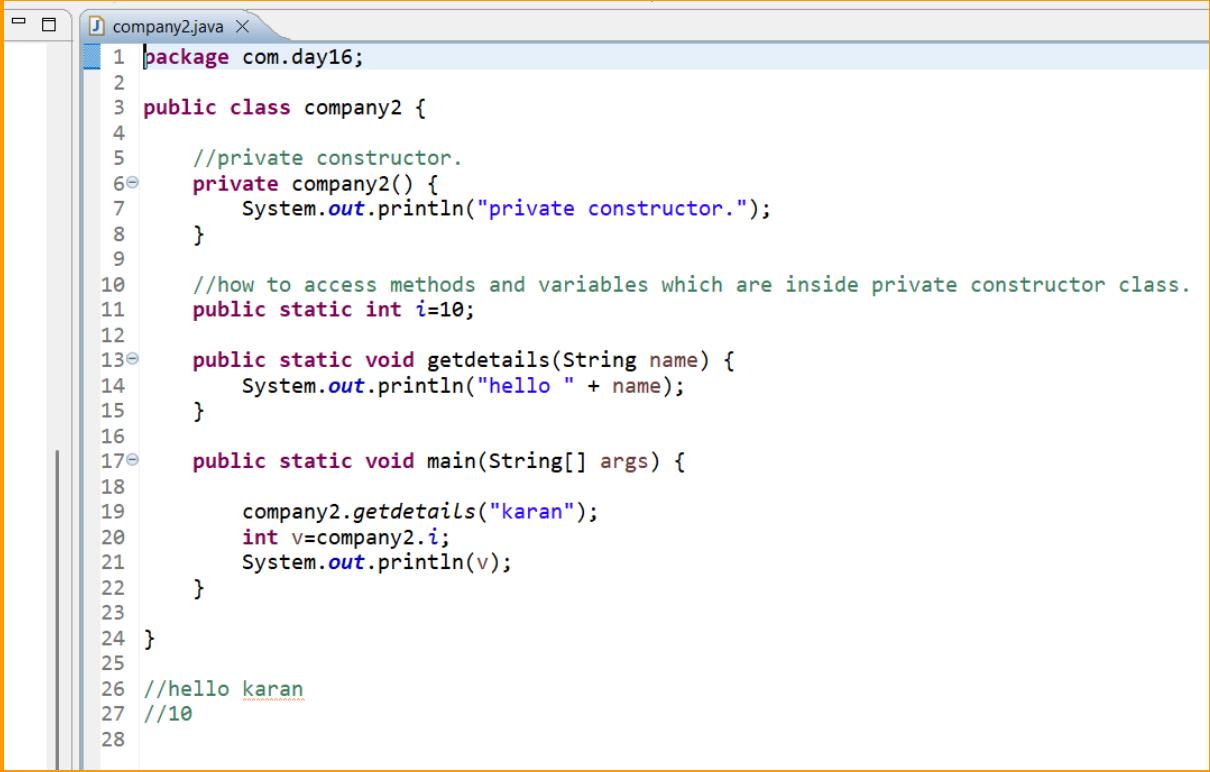
```

company1.java × testcompany1.java
1 package com.day16;
2
3 public class company1 {
4
5     //private constructor.
6     private company1() {
7         System.out.println("private constructor.");
8     }
9
10    //how to access methods and variables which are inside private constructor class.
11    //create everything as static so we dont have to create objects.
12    public static int i=10;
13
14    public static void getdetails(String name) {
15        System.out.println("hello " + name);
16    }
17
18 }
19

testcompany1.java ×
1 package com.day16;
2
3 public class testcompany1 {
4
5     public static void main(String[] args) {
6
7         company1.getdetails("karan");
8         int v=company1.i;
9         System.out.println(v);
10    }
11
12 }
13
14
15 //hello karan
16 //10
17

```

## paste company2



```
1 package com.day16;
2
3 public class company2 {
4
5     //private constructor.
6     private company2() {
7         System.out.println("private constructor.");
8     }
9
10    //how to access methods and variables which are inside private constructor class.
11    public static int i=10;
12
13    public static void getdetails(String name) {
14        System.out.println("hello " + name);
15    }
16
17    public static void main(String[] args) {
18
19        company2.getdetails("karan");
20        int v=company2.i;
21        System.out.println(v);
22    }
23
24 }
25
26 //hello karan
27 //10
28
```

## paste company3, testcompany3

The screenshot shows a Java IDE interface with two code editors. The top editor contains the file `company3.java` with the following code:

```
1 package com.day16;
2
3 public class company3 {
4
5     //private constructor.
6     private company3() {
7         System.out.println("private constructor.");
8     }
9
10    //how to access methods and variables which are inside private constructor class.
11    public static int i=10;
12
13    public static void getdetails(String name) {
14        System.out.println("hello " + name);
15    }
16
17 }
18
19 }
```

The bottom editor contains the file `testcompany3.java` with the following code:

```
1 package com.day16;
2
3 public class testcompany3 {
4
5     public static void main(String[] args) {
6
7         //cant create object of private constructor class.
8         company3 c1=new company3(); //The constructor company3() is not visible
9         c1.getdetails("karan");
10        int v1=c1.i;
11        System.out.println(v1);
12    }
13
14 }
15 }
```

We can have public and private constructor in class-

```
1 package EncapsulationConcept;
2
3 public class Compnay {
4
5
6     private Compnay() {
7
8     }
9
10 public Compnay(int a) {
11
12 }
13
14
15     public static void getRevenue() {
16         System.out.println("get rev");
17     }
18
19
20 }
21
```

paste company4 and test company4-

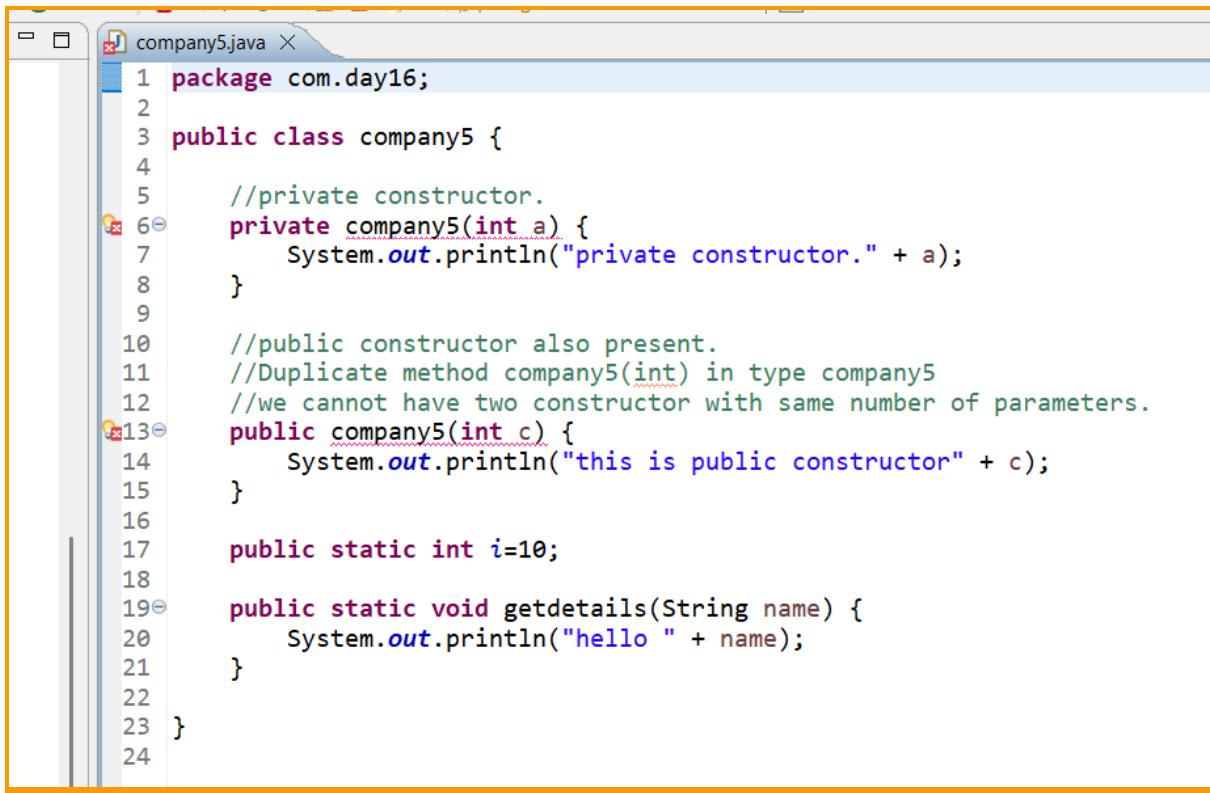
```

company4.java
1 package com.day16;
2
3 public class company4 {
4
5     //private constructor.
6     private company4(int a) {
7         System.out.println("private constructor." + a);
8     }
9
10    //public constructor also present.
11    public company4(String c) {
12        System.out.println("this is public constructor" + c);
13    }
14
15    //how to access methods and variables which are inside private constructor class.
16    //create everything as static so we dont have to create objects.
17    public static int i=10;
18
19    public static void getdetails(String name) {
20        System.out.println("hello " + name);
21    }
22
23 }
24

testcompany4.java
1 package com.day16;
2
3 public class testcompany4 {
4
5     public static void main(String[] args) {
6
7         company4 c2=new company4("lion");
8         c2.getdetails("karan");
9         //warning - The static method getdetails(String) from the type company4 should be accessed in a static way
10        int v2=c2.i;
11        //warning - The static field company4.i should be accessed in a static way
12        System.out.println(v2);
13
14        //change the static value.
15        int s=company4.i=20;
16        System.out.println(s);
17
18        company4.getdetails("giraffe");
19    }
20
21 }
22
23 //this is public constructorlion
24 //hello karan
25 //10
26 //20
27 //hello giraffe
28
29

```

paste company5-

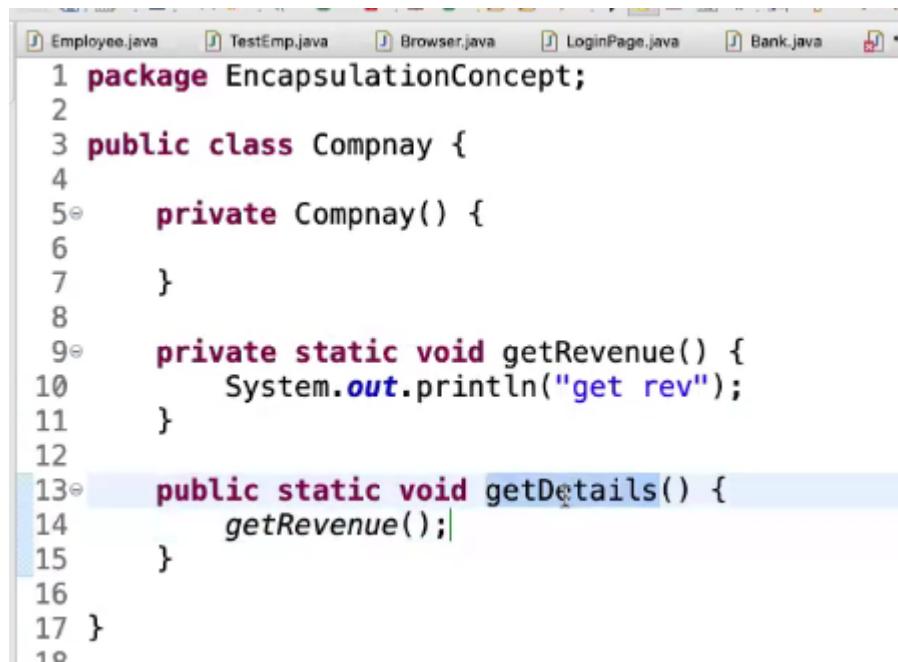


```

1 package com.day16;
2
3 public class company5 {
4
5     //private constructor.
6     private company5(int a) {
7         System.out.println("private constructor." + a);
8     }
9
10    //public constructor also present.
11    //Duplicate method company5(int) in type company5
12    //we cannot have two constructor with same number of parameters.
13    public company5(int c) {
14        System.out.println("this is public constructor" + c);
15    }
16
17    public static int i=10;
18
19    public static void getdetails(String name) {
20        System.out.println("hello " + name);
21    }
22
23}
24

```

If method is also private-



```

1 package EncapsulationConcept;
2
3 public class Compnay {
4
5     private Compnay() {
6
7     }
8
9     private static void getRevenue() {
10        System.out.println("get rev");
11    }
12
13    public static void getDetails() {
14        getRevenue();
15    }
16
17}
18

```

Create public method and encapsulate private inside it.

```
1 package EncapsulationConcept;
2
3
4 public class TestBank {
5
6     public static void main(String[] args) {
7
8         Bank b = new Bank("Pooja", 50, "21212121", "21212121");
9
10        System.out.println(b.getAadharNumber());
11
12        System.out.println(b.getAge());
13
14        b.openAccount();
15
16
17        Compnay.getDetails();
18
19    }
20
21 }
22 }
```

paste company6 and testcompany6-

```
company6.java X
1 package com.day16;
2
3 public class company6 {
4
5     //private constructor.
6     private company6(int a) {
7         System.out.println("private constructor." + a);
8     }
9
10    //private method.
11    private void getdetails(String name) {
12        System.out.println("get details method." + name);
13    }
14
15    private void getaddress() {
16        System.out.println("getaddress method");
17    }
18
19    private static void getphonenumer() {
20        System.out.println("getphonenumer method");
21    }
22
23    private static void getlocation(String location) {
24        System.out.println("getlocation method " + location);
25    }
26
27    //one public method to call all private methods.
28    //encapsulation concept.
29    public static void getalldetails(String name, String location) {
30        //how to access private constructor inside public static method.
31        company6 c1=new company6(10);
32        c1.getdetails(name);
33        c1.getaddress();
34        getphonenumer();
35        getlocation(location);
36    }
37
38 }
39
```

```

1 package com.day16;
2
3 public class testcompany6 {
4
5     public static void main(String[] args) {
6
7         company6.getAllDetails("karan", "giraffe");
8     }
9
10 }
11
12 //private constructor.10
13 //get details method.karan
14 //getaddress method
15 //getphonenumer
16 //getlocation method giraffe
17
18

```

Static variable can be accessed inside constructor-

```

4
5 // private class variables
6 private String name;
7 private int age;
8 private double salary;
9 private boolean isPerm;
10 static int wheels = 4;
11
12 //public const.. is also like setter
13 public Employee(String name, int age, double salary, boolean isPerm) {
14     this.name = name;
15     this.age = age;
16     this.salary = salary;
17     this.isPerm = isPerm;
18     Employee.wheels = 5;
19 }

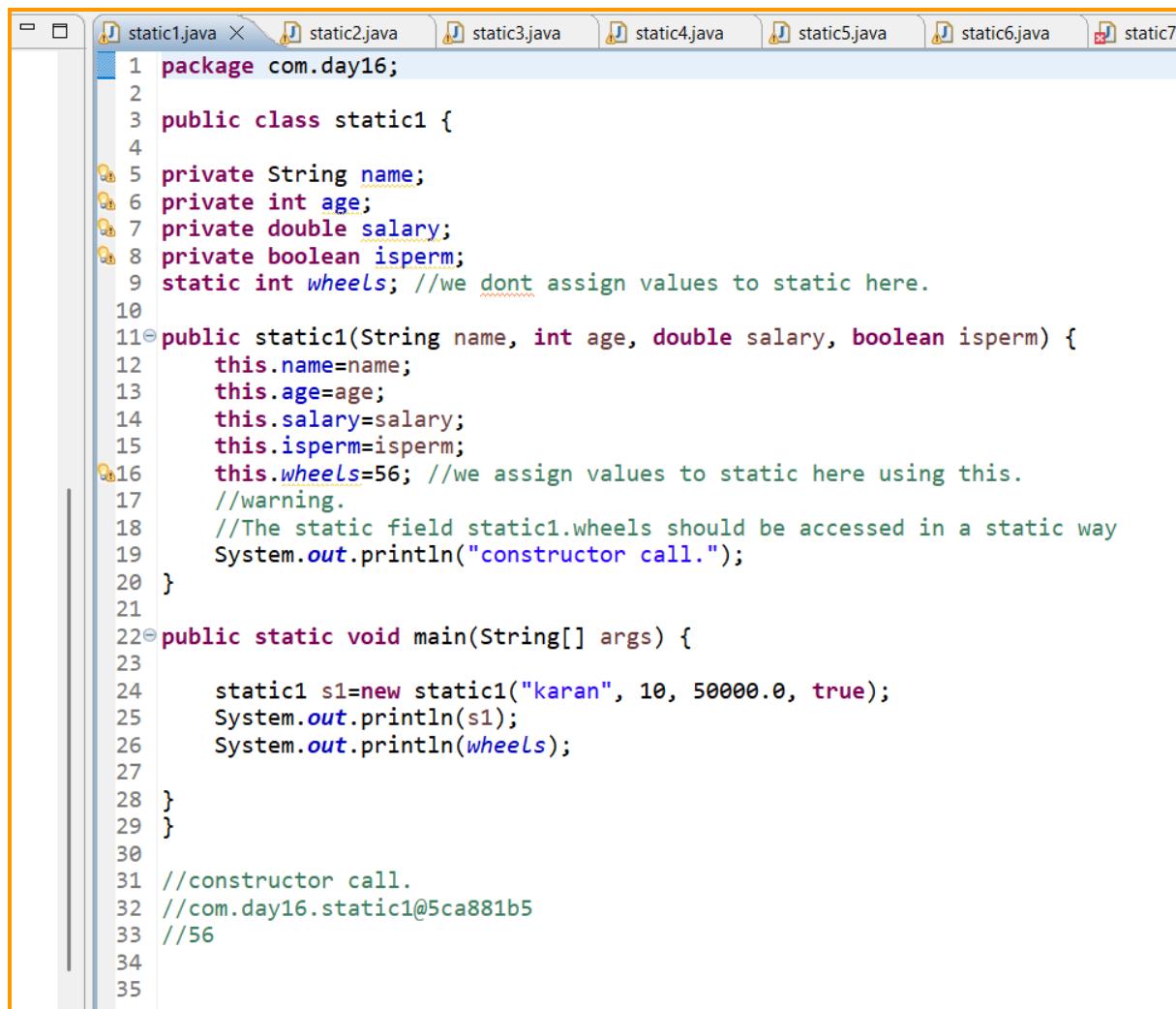
```

But mandatory to give some value to it, else error.

//Syntax error, insert "VariableDeclarators" to complete LocalVariableDeclaration

paste static 1 to 7-

different combinations tried with static.



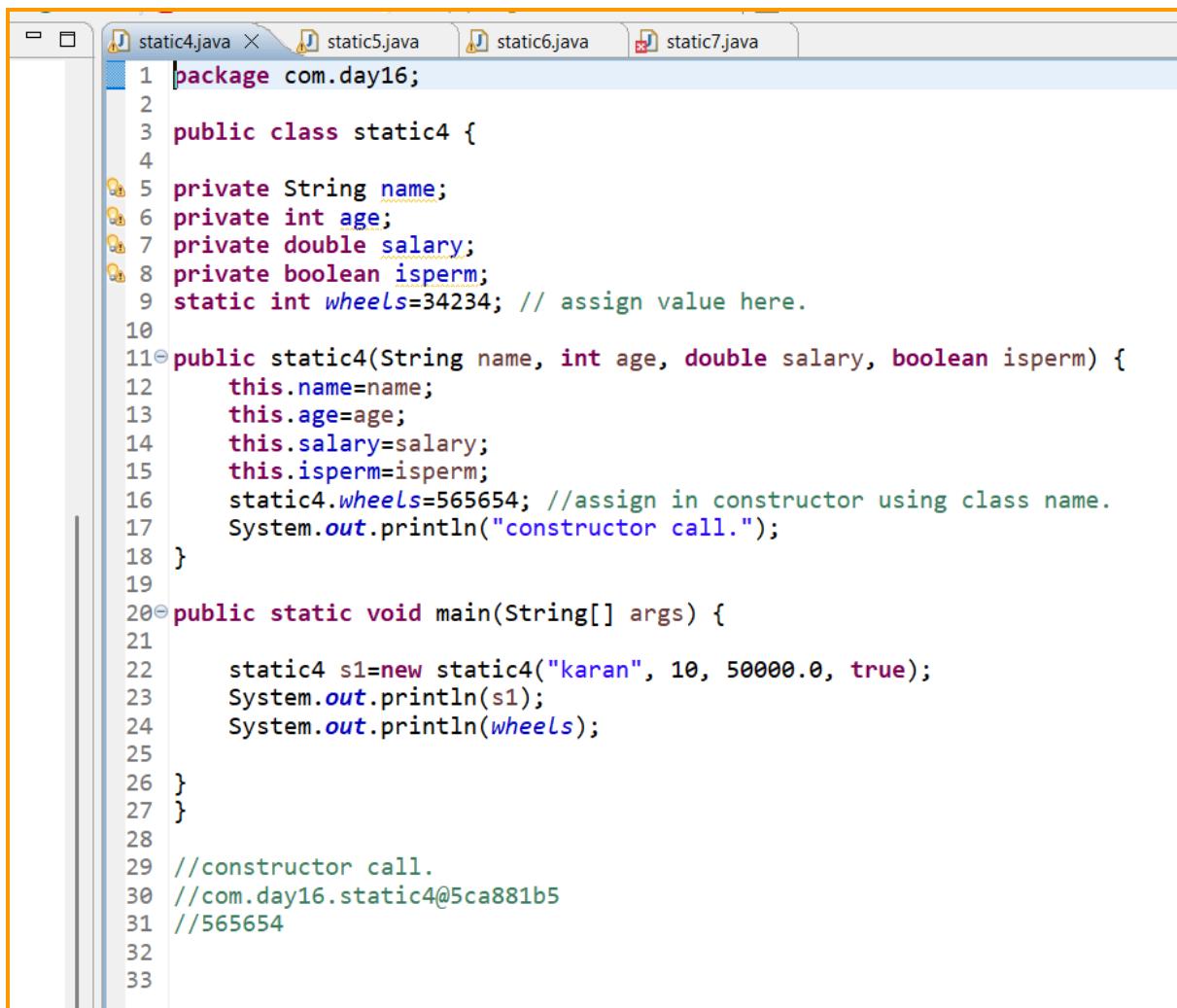
```
1 package com.day16;
2
3 public class static1 {
4
5     private String name;
6     private int age;
7     private double salary;
8     private boolean isperm;
9     static int wheels; //we dont assign values to static here.
10
11    public static1(String name, int age, double salary, boolean isperm) {
12        this.name=name;
13        this.age=age;
14        this.salary=salary;
15        this.isperm=isperm;
16        this.wheels=56; //we assign values to static here using this.
17        //warning.
18        //The static field static1.wheels should be accessed in a static way
19        System.out.println("constructor call.");
20    }
21
22    public static void main(String[] args) {
23
24        static1 s1=new static1("karan", 10, 50000.0, true);
25        System.out.println(s1);
26        System.out.println(wheels);
27
28    }
29 }
30
31 //constructor call.
32 //com.day16.static1@5ca881b5
33 //56
34
35
```

The screenshot shows a Java code editor with the file `static2.java` open. The code defines a class `static2` with private instance variables `name`, `age`, `salary`, and `isperm`, and a static field `wheels`. It includes a constructor that initializes these variables and prints the value of `wheels`. The code is annotated with comments explaining the usage of static fields and constructors.

```
1 package com.day16;
2
3 public class static2 {
4
5     private String name;
6     private int age;
7     private double salary;
8     private boolean isperm;
9     static int wheels=5; //assign value to static here.
10
11    public static2(String name, int age, double salary, boolean isperm) {
12        this.name=name;
13        this.age=age;
14        this.salary=salary;
15        this.isperm=isperm;
16        this.wheels=56; //reassign in constructor using this.
17        //warning.
18        //The static field static1.wheels should be accessed in a static way
19        System.out.println("constructor call.");
20    }
21
22    public static void main(String[] args) {
23
24        static2 s1=new static2("karan", 10, 50000.0, true);
25        System.out.println(s1);
26        System.out.println(wheels);
27    }
28
29 }
30
31 //constructor call.
32 //com.day16.static2@5ca881b5
33 //56
```

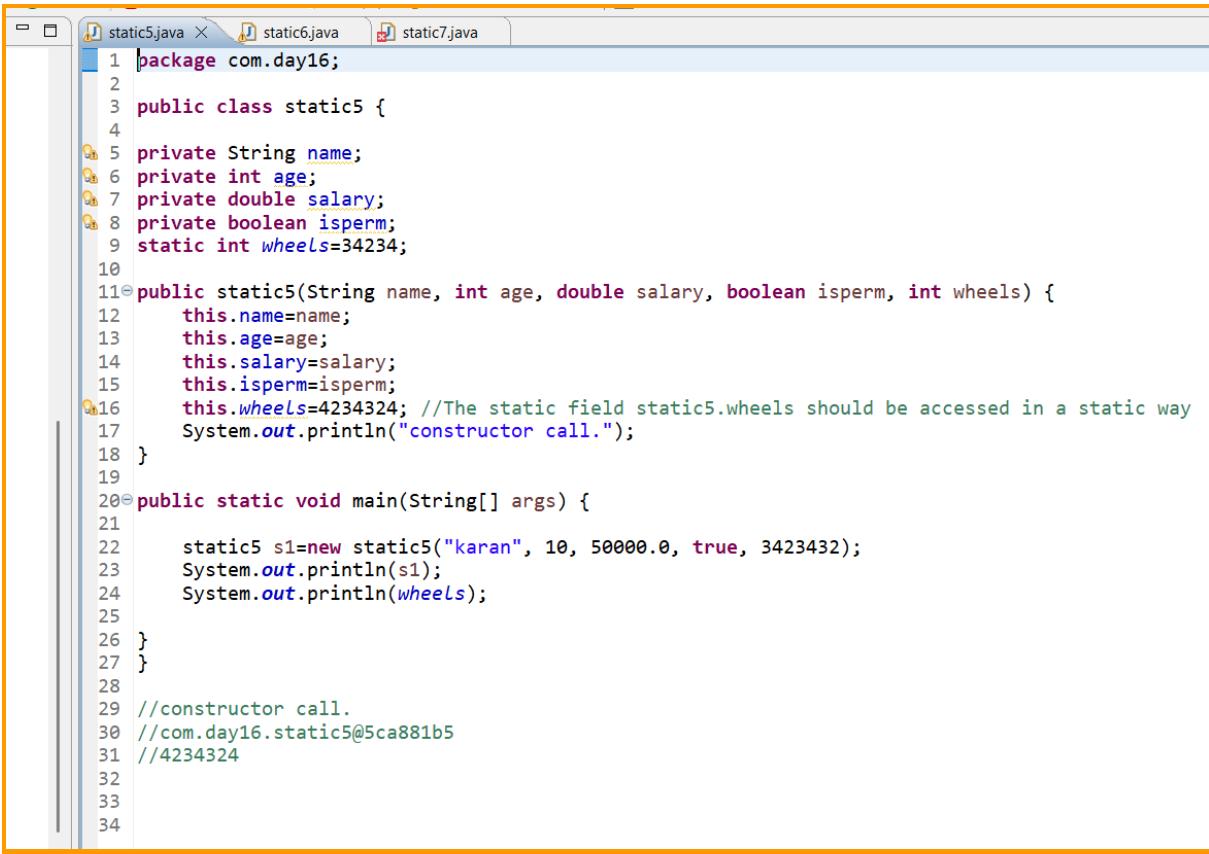
The screenshot shows a Java code editor with the file `static3.java` open. The code defines a class `static3` with private instance variables `name`, `age`, `salary`, and `isperm`. It includes a constructor that initializes these variables and sets a static variable `wheels` to 56. The `main` method creates an object `s1` and prints its values and the value of `wheels`. The code is annotated with comments explaining the usage of static variables.

```
1 package com.day16;
2
3 public class static3 {
4
5     private String name;
6     private int age;
7     private double salary;
8     private boolean isperm;
9     static int wheels; //dont assign value here.
10
11    public static3(String name, int age, double salary, boolean isperm) {
12        this.name=name;
13        this.age=age;
14        this.salary=salary;
15        this.isperm=isperm;
16        static3.wheels=56; //assign in constructor using class name.
17        System.out.println("constructor call.");
18    }
19
20    public static void main(String[] args) {
21
22        static3 s1=new static3("karan", 10, 50000.0, true);
23        System.out.println(s1);
24        System.out.println(wheels);
25
26    }
27 }
28 //constructor call.
29 //com.day16.static3@5ca881b5
30 //56
31
32
```



The screenshot shows a Java code editor with the file `static4.java` open. The code defines a class `static4` with private instance variables `name`, `age`, `salary`, and `isperm`, and a static variable `wheels`. It includes a constructor that initializes the instance variables and sets `wheels` to 565654. The `main` method creates an object `s1` and prints its `wheels` value. The code is annotated with comments explaining the usage of static variables.

```
1 package com.day16;
2
3 public class static4 {
4
5     private String name;
6     private int age;
7     private double salary;
8     private boolean isperm;
9     static int wheels=34234; // assign value here.
10
11    public static4(String name, int age, double salary, boolean isperm) {
12        this.name=name;
13        this.age=age;
14        this.salary=salary;
15        this.isperm=isperm;
16        static4.wheels=565654; //assign in constructor using class name.
17        System.out.println("constructor call.");
18    }
19
20    public static void main(String[] args) {
21
22        static4 s1=new static4("karan", 10, 50000.0, true);
23        System.out.println(s1);
24        System.out.println(wheels);
25
26    }
27 }
28
29 //constructor call.
30 //com.day16.static4@5ca881b5
31 //565654
32
33
```



The screenshot shows a Java code editor with a file named static5.java open. The code defines a class static5 with private instance variables name, age, salary, and isperm, and a static variable wheels. It includes a constructor that initializes these variables and prints a message. The main method creates an object s1 and prints its name and the value of wheels. A note in the code indicates that wheels should be accessed in a static way.

```
1 package com.day16;
2
3 public class static5 {
4
5     private String name;
6     private int age;
7     private double salary;
8     private boolean isperm;
9     static int wheels=34234;
10
11    public static5(String name, int age, double salary, boolean isperm, int wheels) {
12        this.name=name;
13        this.age=age;
14        this.salary=salary;
15        this.isperm=isperm;
16        this.wheels=4234324; //The static field static5.wheels should be accessed in a static way
17        System.out.println("constructor call.");
18    }
19
20    public static void main(String[] args) {
21
22        static5 s1=new static5("karan", 10, 50000.0, true, 3423432);
23        System.out.println(s1);
24        System.out.println(wheels);
25
26    }
27 }
28
29 //constructor call.
30 //com.day16.static5@5ca881b5
31 //4234324
32
33
34
```

```

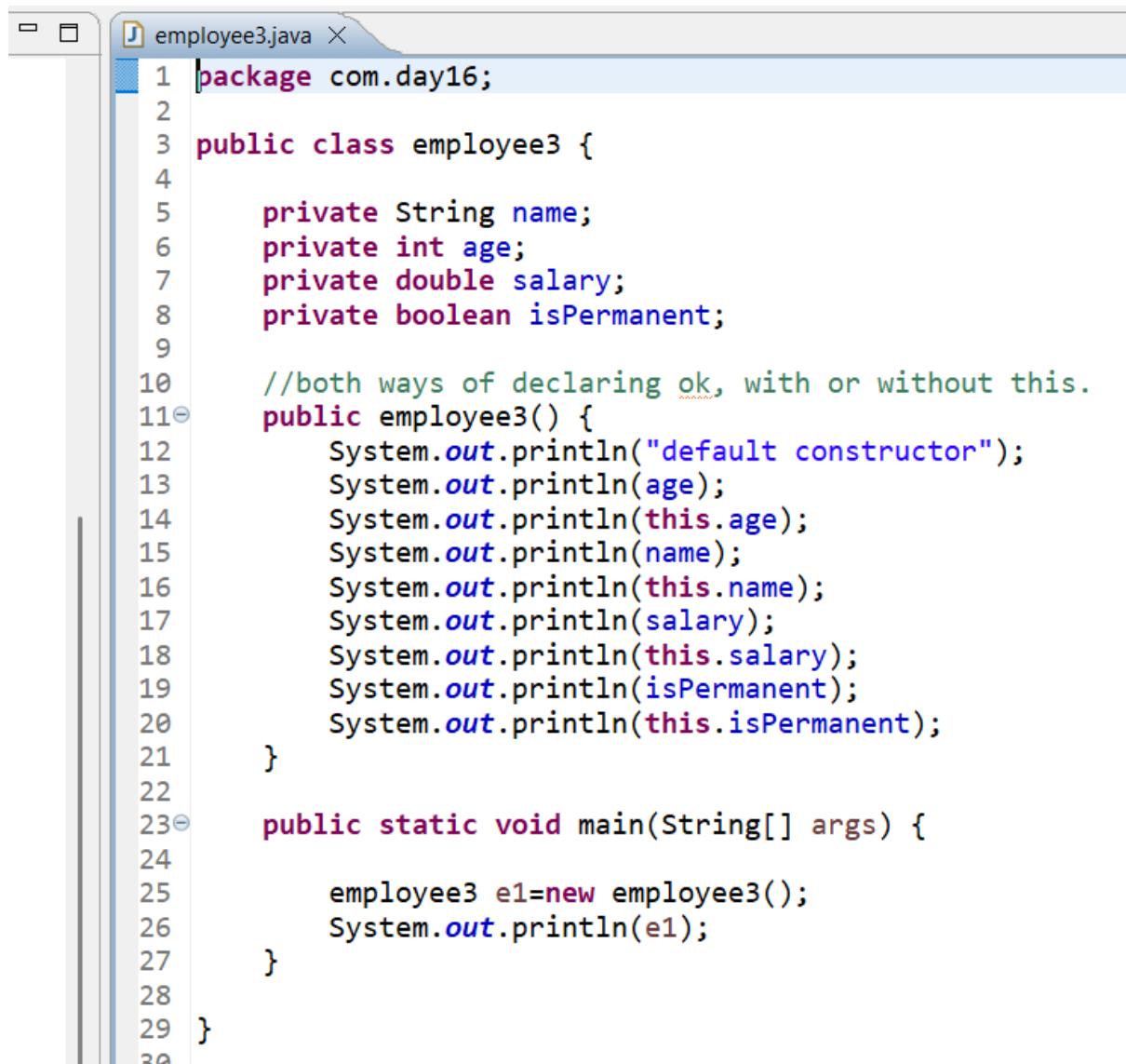
static6.java
1 package com.day16;
2
3 public class static6 {
4
5     private String name;
6     private int age;
7     private double salary;
8     private boolean isperm;
9     static int wheels=34234;
10
11    //what ever is in constructor that is the last value of static so it gets printed.
12
13    public static6(String name, int age, double salary, boolean isperm, int wheels) {
14        this.name=name;
15        this.age=age;
16        this.salary=salary;
17        this.isperm=isperm;
18        static6.wheels=4234324;
19        System.out.println("constructor call.");
20    }
21
22    public static void main(String[] args) {
23
24        static6 s1=new static6("karan", 10, 50000.0, true, 3423432);
25        System.out.println(s1);
26        System.out.println(wheels);
27
28    }
29 }
30
31 //constructor call.
32 //com.day16.static6@5ca881b5
33 //4234324
34
35
36

static7.java
1 package com.day16;
2
3 public class static7 {
4
5     private String name;
6     private int age;
7     private double salary;
8     private boolean isperm;
9     static int wheels=34234;
10
11    public static7(String name, int age, double salary, boolean isperm) {
12        this.name=name;
13        this.age=age;
14        this.salary=salary;
15        this.isperm=isperm;
16        static7.wheels; //Syntax error, insert "VariableDeclarators" to complete LocalVariableDeclaration
17        System.out.println("constructor call.");
18    }
19
20    public static void main(String[] args) {
21
22        static7 s1=new static7("karan", 10, 50000.0, true);
23        System.out.println(s1);
24        System.out.println(wheels);
25
26    }
27 }
28
29
30
31

```

To get the default values use default constructor-  
1.37.52

paste employee3



The screenshot shows a Java code editor window with the file "employee3.java" open. The code defines a class "employee3" with private fields for name, age, salary, and isPermanent. It includes a constructor that prints the values of these fields using System.out.println. The main method creates an instance of employee3 and prints it.

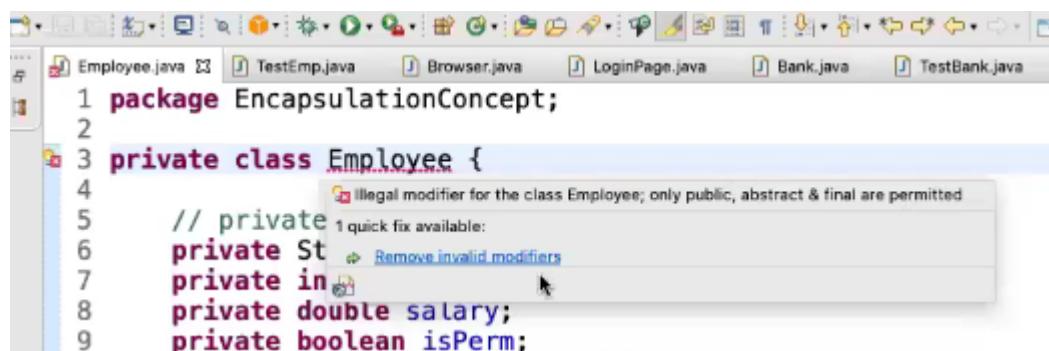
```
1 package com.day16;
2
3 public class employee3 {
4
5     private String name;
6     private int age;
7     private double salary;
8     private boolean isPermanent;
9
10    //both ways of declaring ok, with or without this.
11    public employee3() {
12        System.out.println("default constructor");
13        System.out.println(age);
14        System.out.println(this.age);
15        System.out.println(name);
16        System.out.println(this.name);
17        System.out.println(salary);
18        System.out.println(this.salary);
19        System.out.println(isPermanent);
20        System.out.println(this.isPermanent);
21    }
22
23    public static void main(String[] args) {
24
25        employee3 e1=new employee3();
26        System.out.println(e1);
27    }
28
29 }
```

```

28
29 }
30
31 //default constructor
32 //0
33 //0
34 //null
35 //null
36 //0.0
37 //0.0
38 //false
39 //false
40 //com.day16.employee3@5ca881b5
41
42

```

## Private class not allowed-



for static variable no need to assign values.

default value will be taken for that variable's data type.

**paste static8-**

```
static8.java X
1 package com.day16;
2
3 public class static8 {
4
5     private String name;
6     private int age;
7     private double salary;
8     private boolean isPermanent;
9     private static int wheels;
10    private static String type;
11    private static boolean flag;
12    private static double number;
13
14    //both ways of declaring ok, with or without this.
15    //print the static variables default value.
16    public static8() {
17        System.out.println("default constructor");
18        System.out.println(age);
19        System.out.println(this.age);
20        System.out.println(name);
21        System.out.println(this.name);
22        System.out.println(salary);
23        System.out.println(this.salary);
24        System.out.println(isPermanent);
25        System.out.println(this.isPermanent);
26        System.out.println(wheels);
27        System.out.println(this.wheels); //The static field static8.wheels should be accessed in a static way
28        System.out.println(type);
29        System.out.println(flag);
30        System.out.println(this.flag); //The static field static8.flag should be accessed in a static way
31        System.out.println(number);
32        System.out.println(this.number); //The static field static8.number should be accessed in a static way
33        System.out.println(this.number); //The static field static8.number should be accessed in a static way
34    }
35
36    public static void main(String[] args) {
37
38        static8 e1=new static8();
39        System.out.println(e1);
40        System.out.println(wheels);
41        System.out.println(type);
42        System.out.println(flag);
43        System.out.println(number);
44    }
}
```

```
12     System.out.println("bug");
13     System.out.println(number);
14 }
15
16 }
17
18 //default constructor
19 //0
20 //0
21 //null
22 //null
23 //0.0
24 //0.0
25 //false
26 //false
27 //0
28 //0
29 //null
30 //null
31 //false
32 //false
33 //0.0
34 //0.0
35 //com.day16.static8@5ca881b5
36 //0
37 //null
38 //false
39 //0.0
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
```

```
66 //raise
67 //0.0
68
69
70
71
72
73
```

