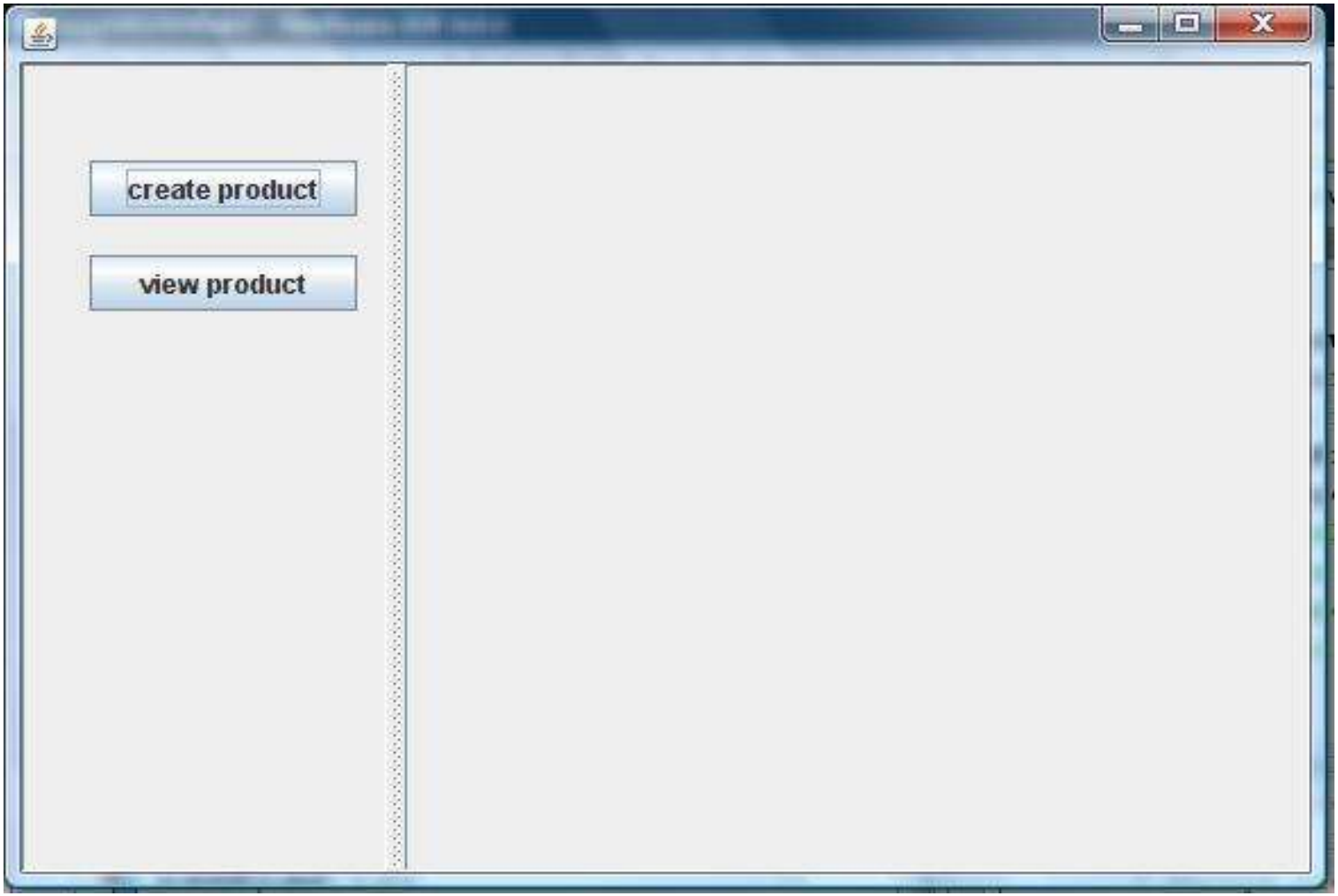
INFO 5100 : Application Engineering and Development

Lab 1 : Class + Object + Relationships

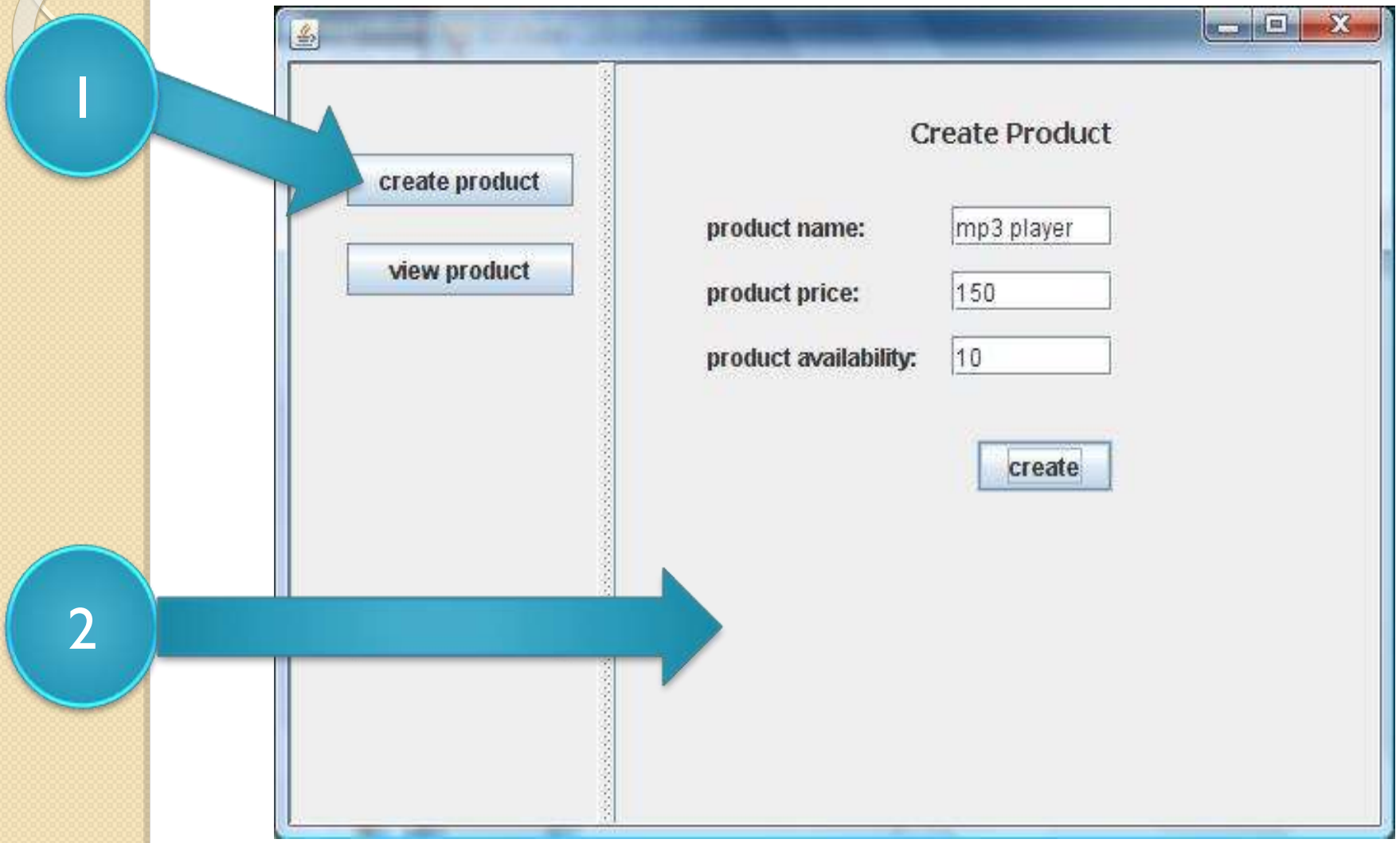
Objective

- Demonstrate how to create a java/swing application
- How to define java classes
- How to create and populate java objects
- How to pass data between from the Jframe to JPanels

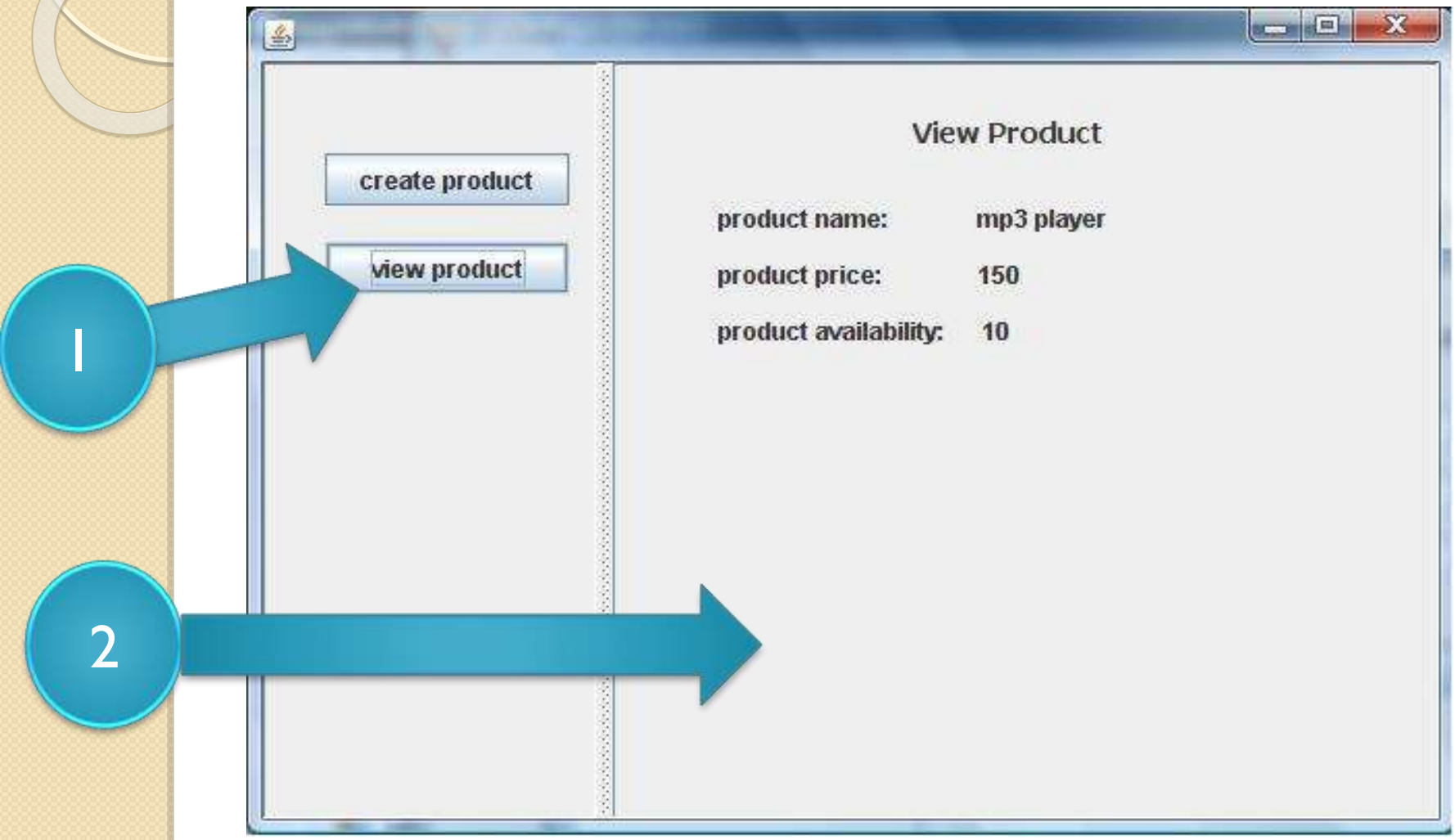
Output Application 1



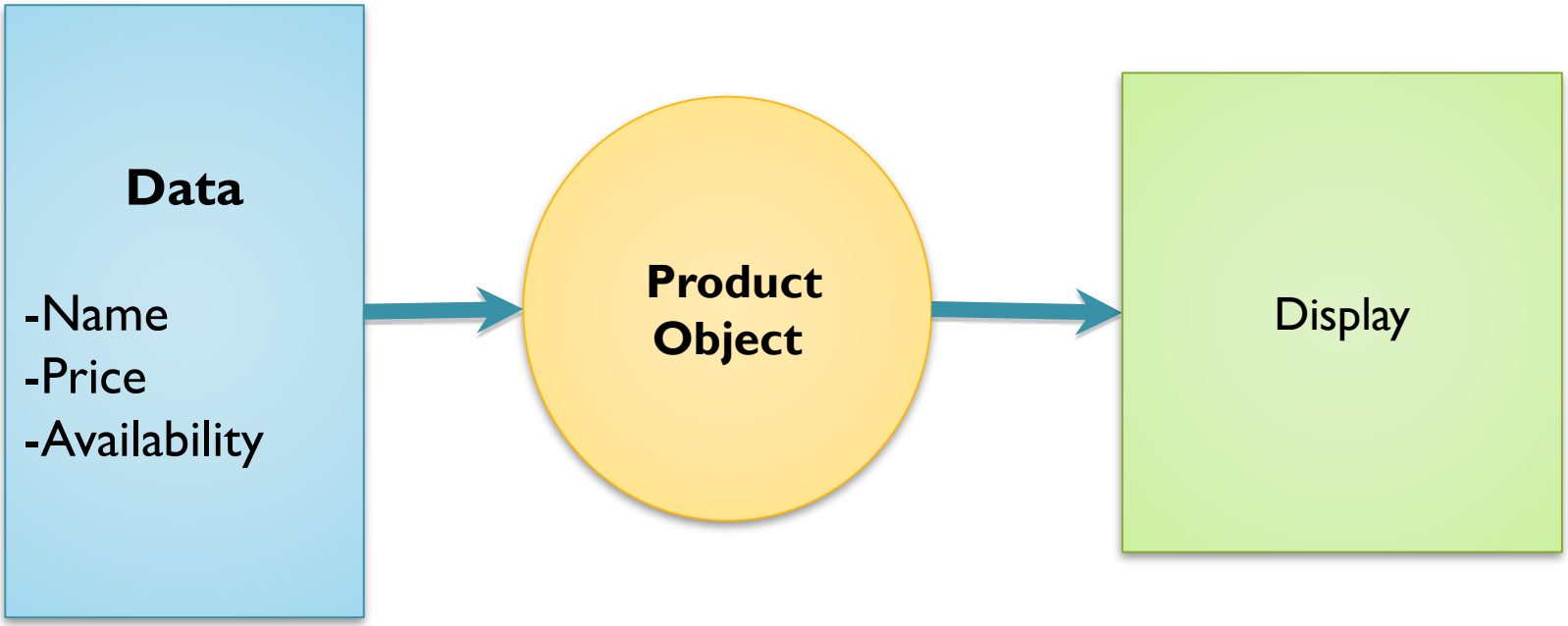
Output Step 1



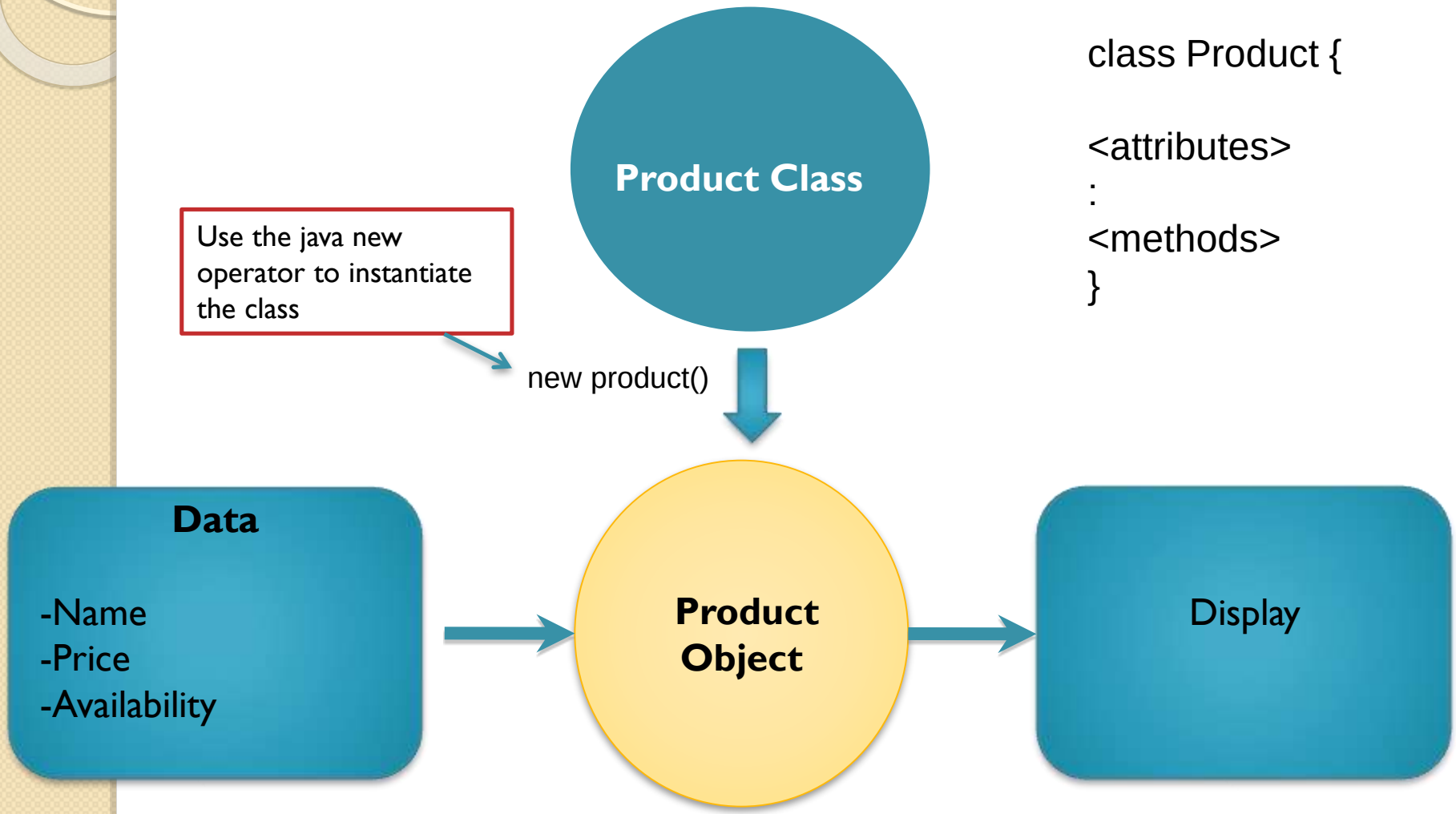
Output Step 2



How to create and move data in and out of objects?



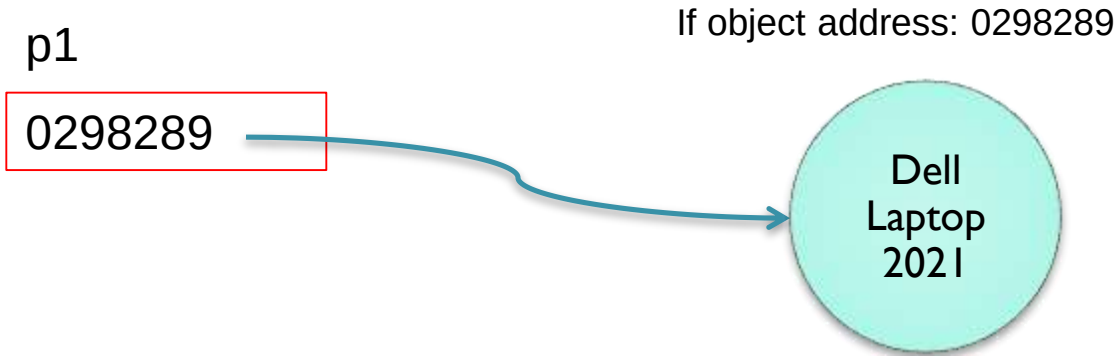
The approach:




```
class Product {

<attributes>
:
<methods>
}
```

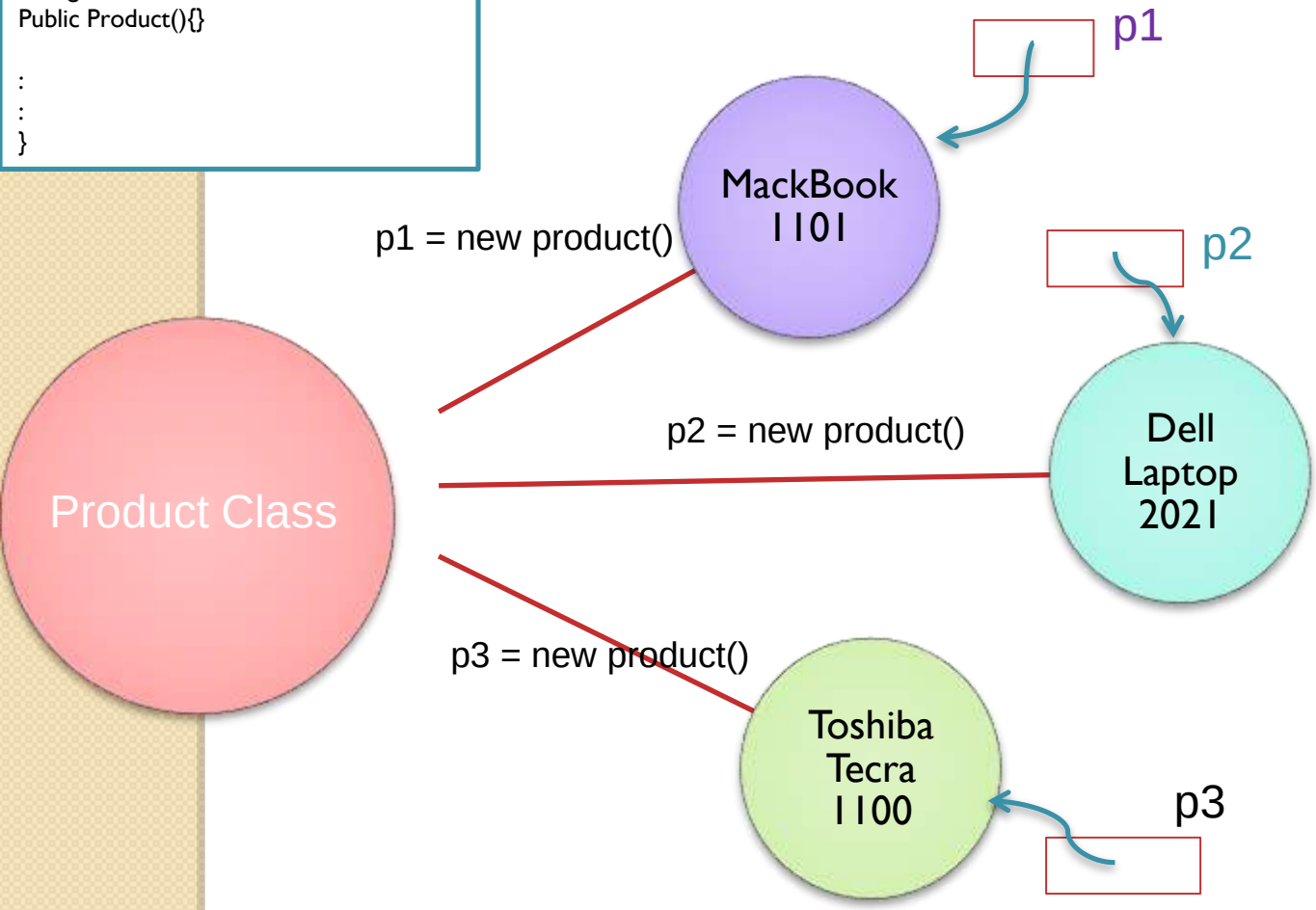
Java Reference variables

1. When we create things we need to keep track of them. If not we would never be able to find them.
2. A reference variable has a name. We use the name to find the variable and then look inside to find information about where the object is
3. Memory space (a place holder) for keeping track of an object but it is not an object in itself
4. An object has a numeric address of where it can be found
5. We save the address inside the reference variable as a way to get hold of the object



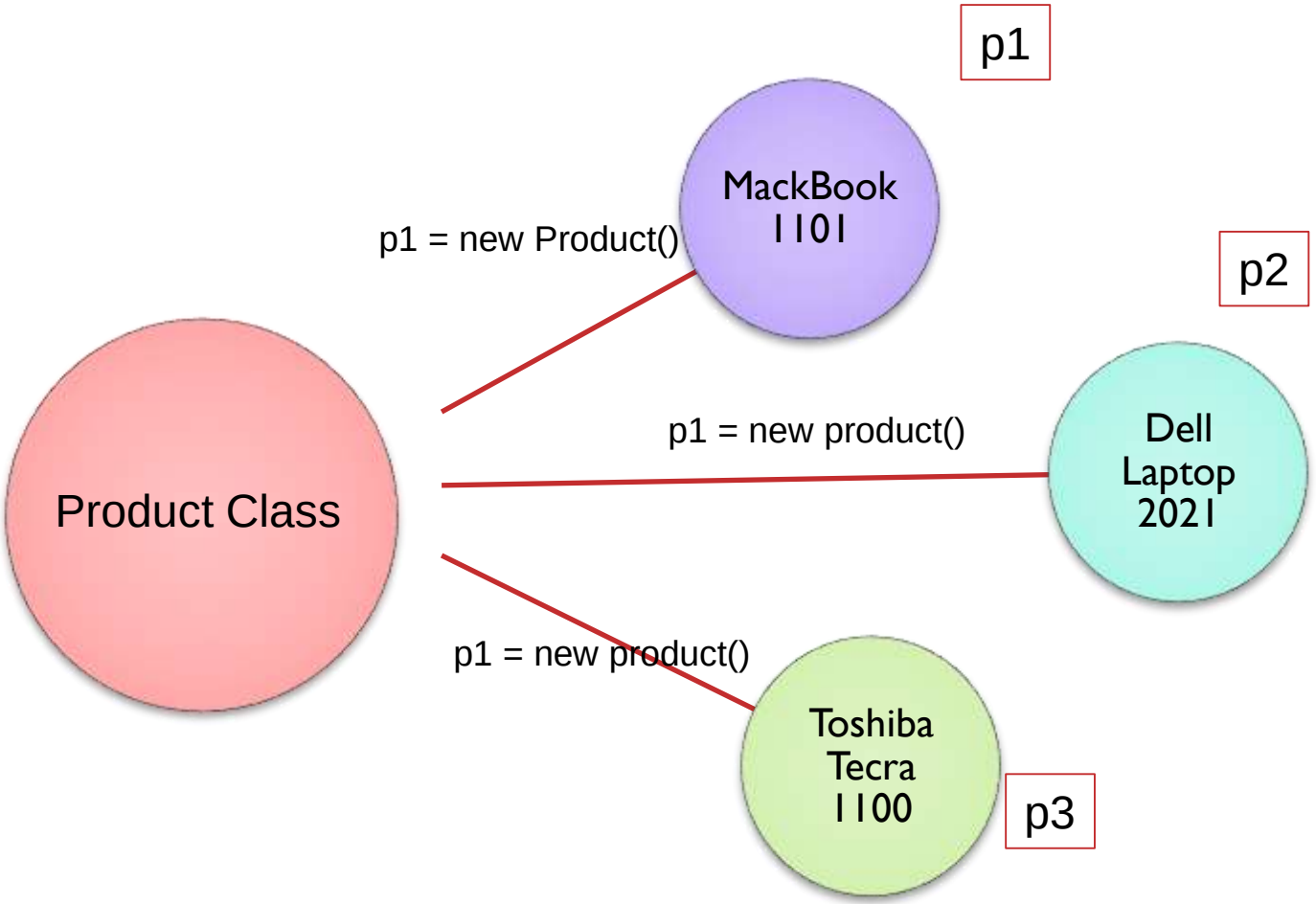
Java Reference variables

```
//example class definition Public class
Product {

String name;
Public Product(){}

:
:
}
```

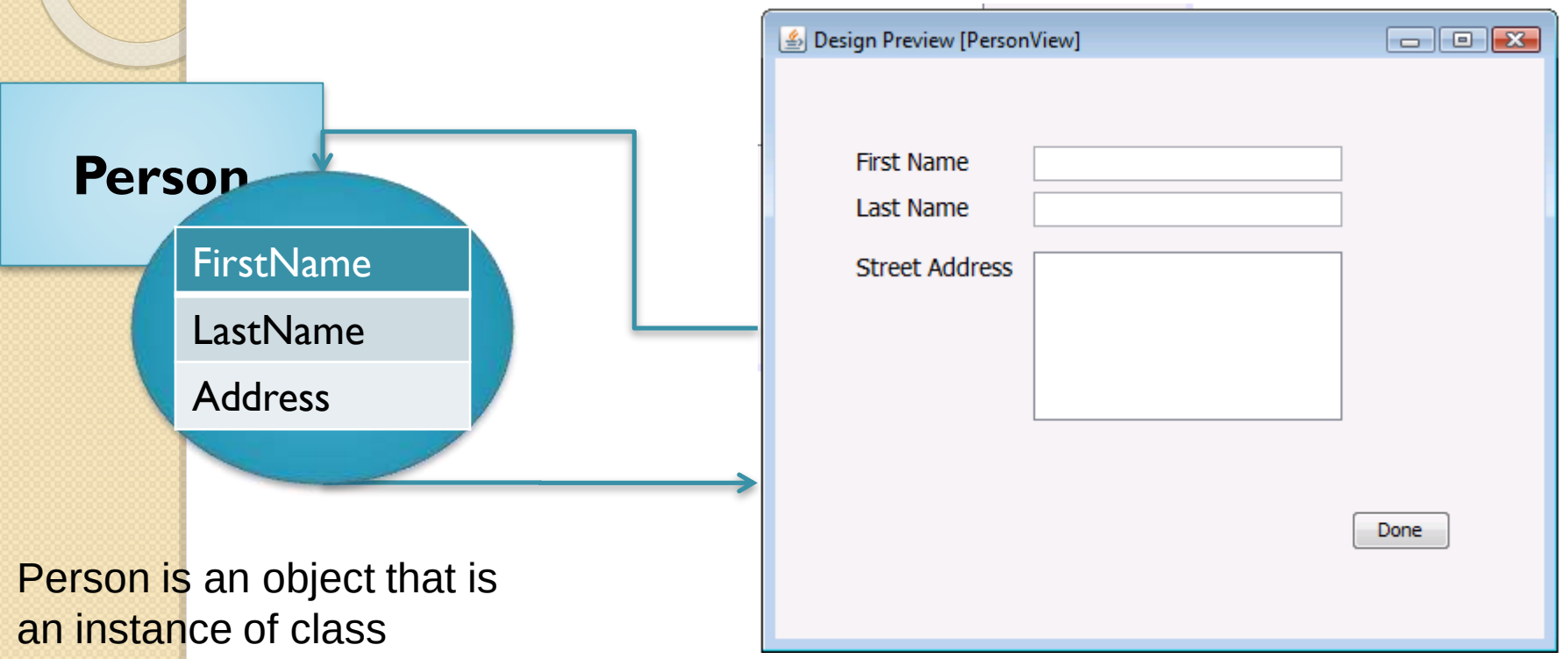



```
//Example main program:

public class Main {


    public static void main(String[] args) {
        // TODO code application logic
here

        Product p1 = new Product();
        p1.setName("MacBook ...");
        Product p2 = new Product();
        p2.setName("Dell ...");
        Product p3 = new Product();
        p3.setName("Toshiba ...");

    }
}
```

Instantiation: From classes to objects

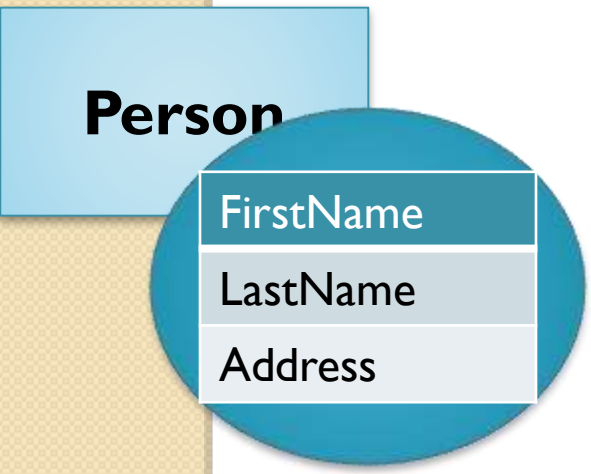


Connecting components to UI



Person is an object that is an instance of class Person. Must be created using the new operator

Data attributes are accessed through their interfaces

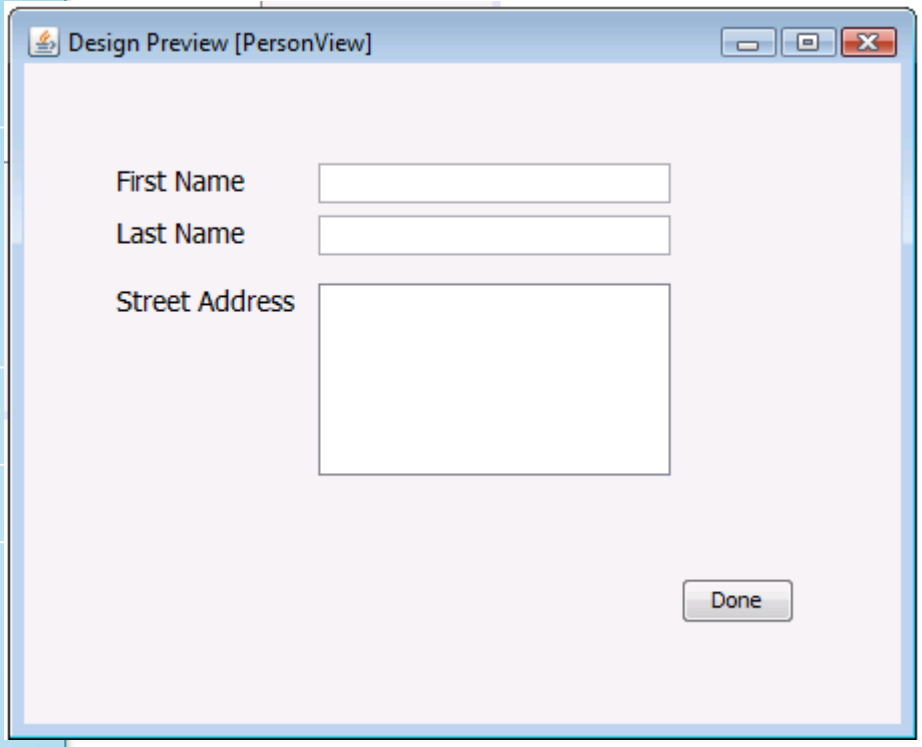



```
class Person {

    lastName: String
    firstName: String
    address:    String

    getLastName();
    getFirstName();
    getFullName();
    getAddress();

    setLastName()
    setFirstName()
    setAddress();

    }
```

Form fields correspond to data attributes

Person

```
class Person {

    lastName: String
    firstName: String
    address:    String

    getLastName();
    getFirstName();
    getFullName();
    getAddress();

    setLastName()
    setFirstName()
    setAddress();

    }
```



Form fields have names

Person

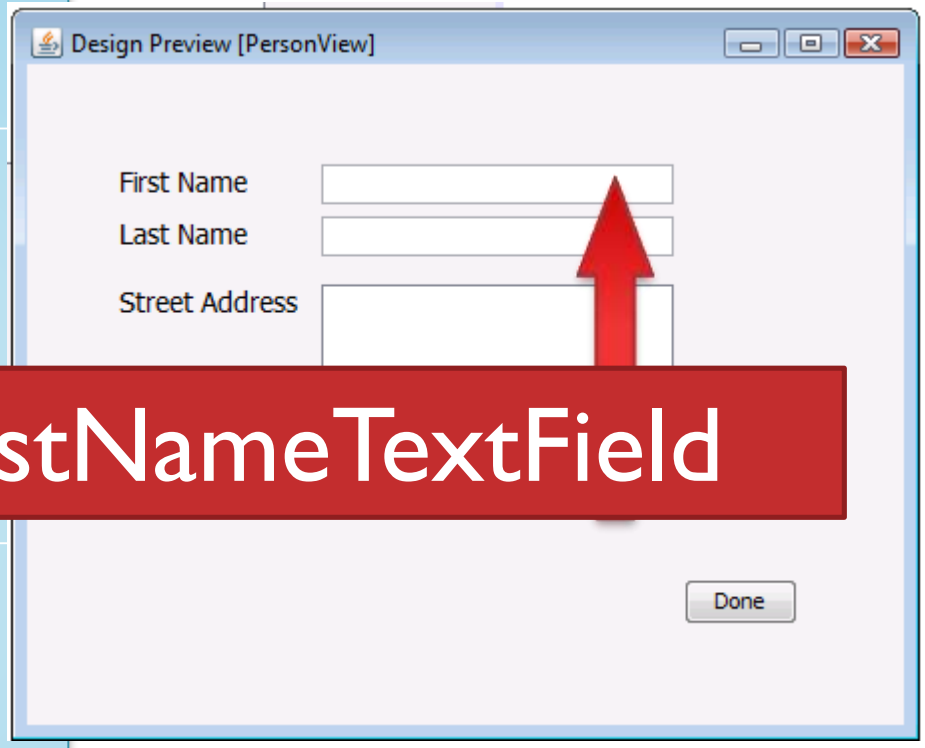
class Person {

lastName: String
firstName: String
address: String

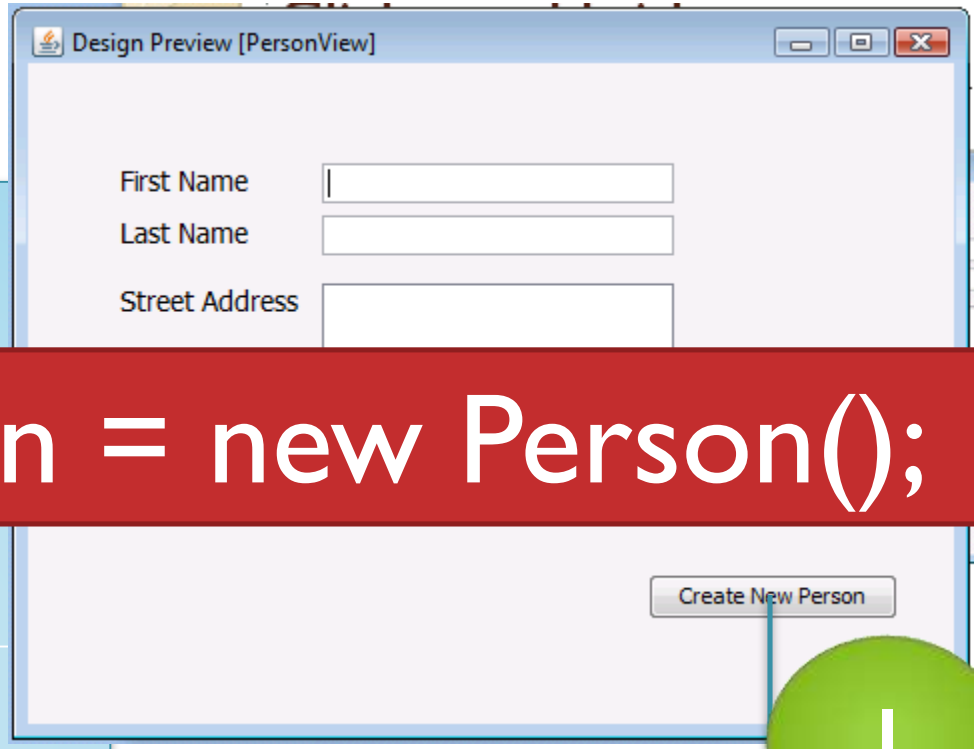
getFullName();
getAddress();

setLastName()
setFirstName()
setAddress();

}



Field name is FirstNameTextField



2

Person person = new Person();

Person

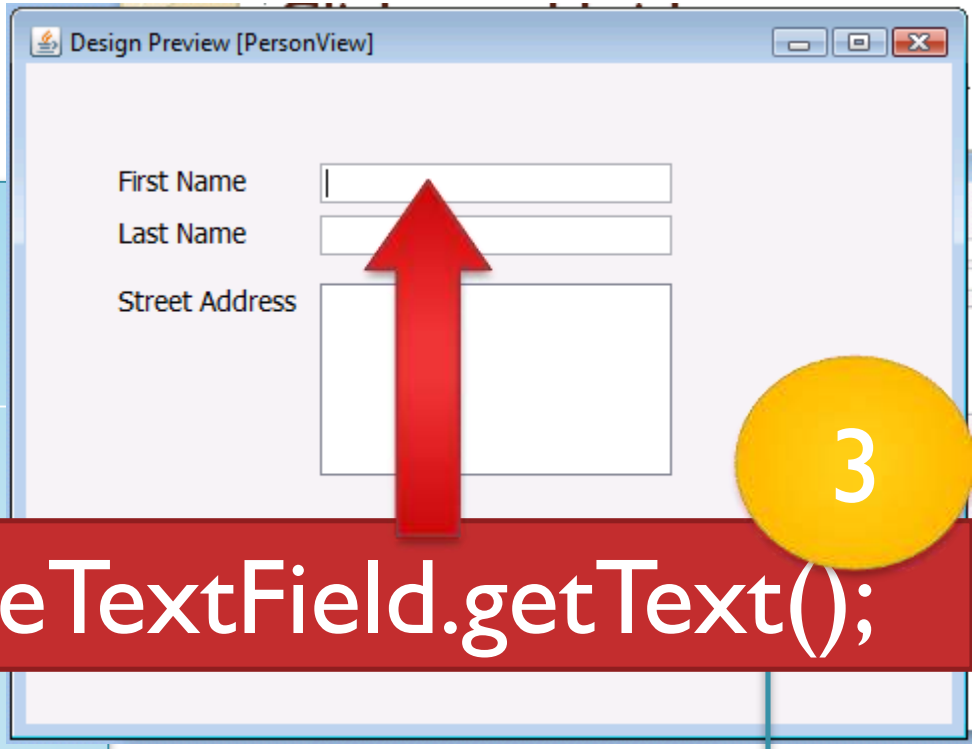
```
class Person {
    ...ess:   String

    getLastName();
    getFirstName();
    getFullName();
    getAddress();

    setLastName()
    setFirstName()
    setAddress();

}
```

1

FirstName
LastName
Address

person

```
Person p = new Person();
String fn = firstnameTextField.getText();
p.setFirstName(fn);
....
```



3

class Person {

lastName: String
firstName: String

getLastName();
getFirstName();
getFullName();
getAddress();

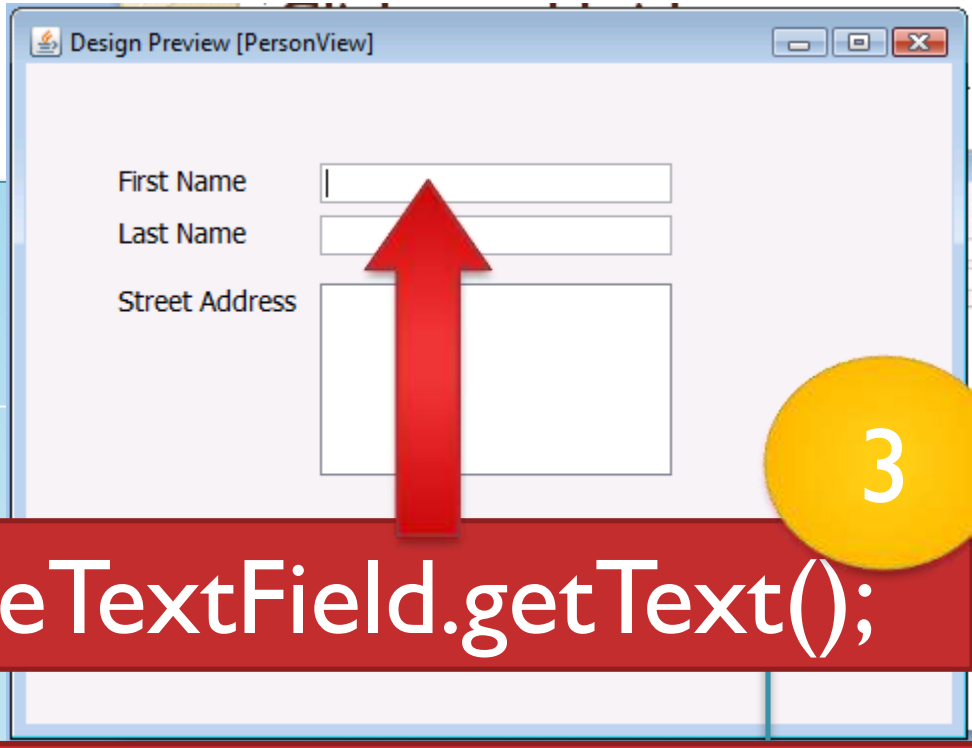
setLastName()
setFirstName()
setAddress();

}

String fn = FirstNameTextField.getText();

person

```
Person p = new Person();
String fn = firstnameTextField.getText();
p.setFirstName(fn);
….
```




```
class Person {

    lastName: String
    firstName: String

    getLastName();

    setAddress();

}
```

3

String fn = FirstNameTextField.getText();

person.setFirstName(fn);

4

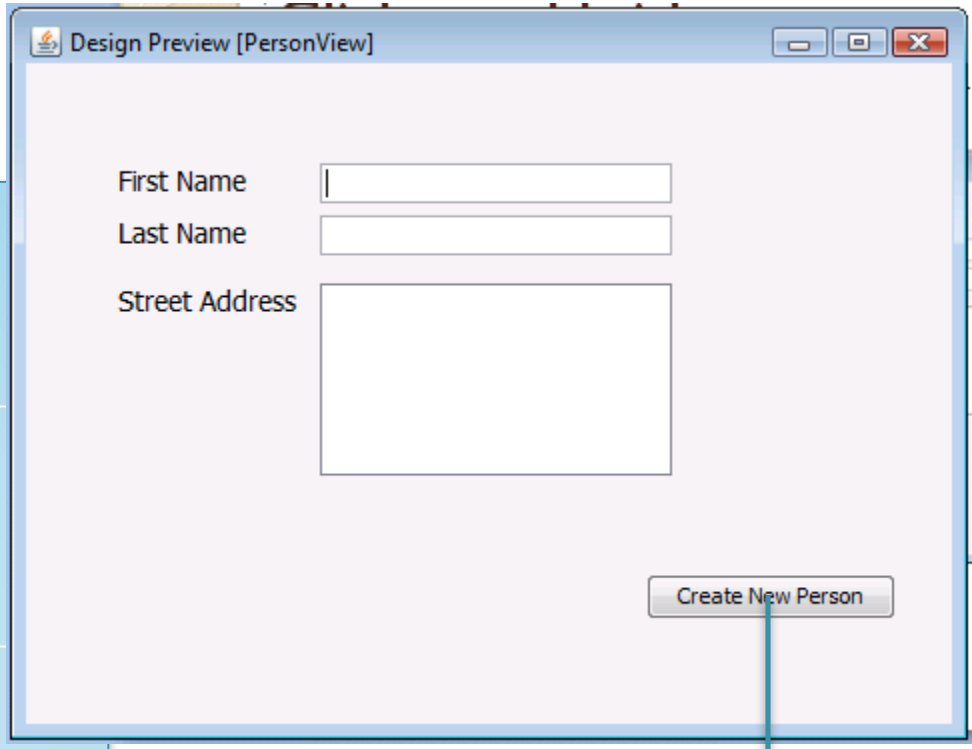
FirstName
LastName
Address

person

```
Person p = new Person();
String fn = firstnameTextField.getText();
....
```

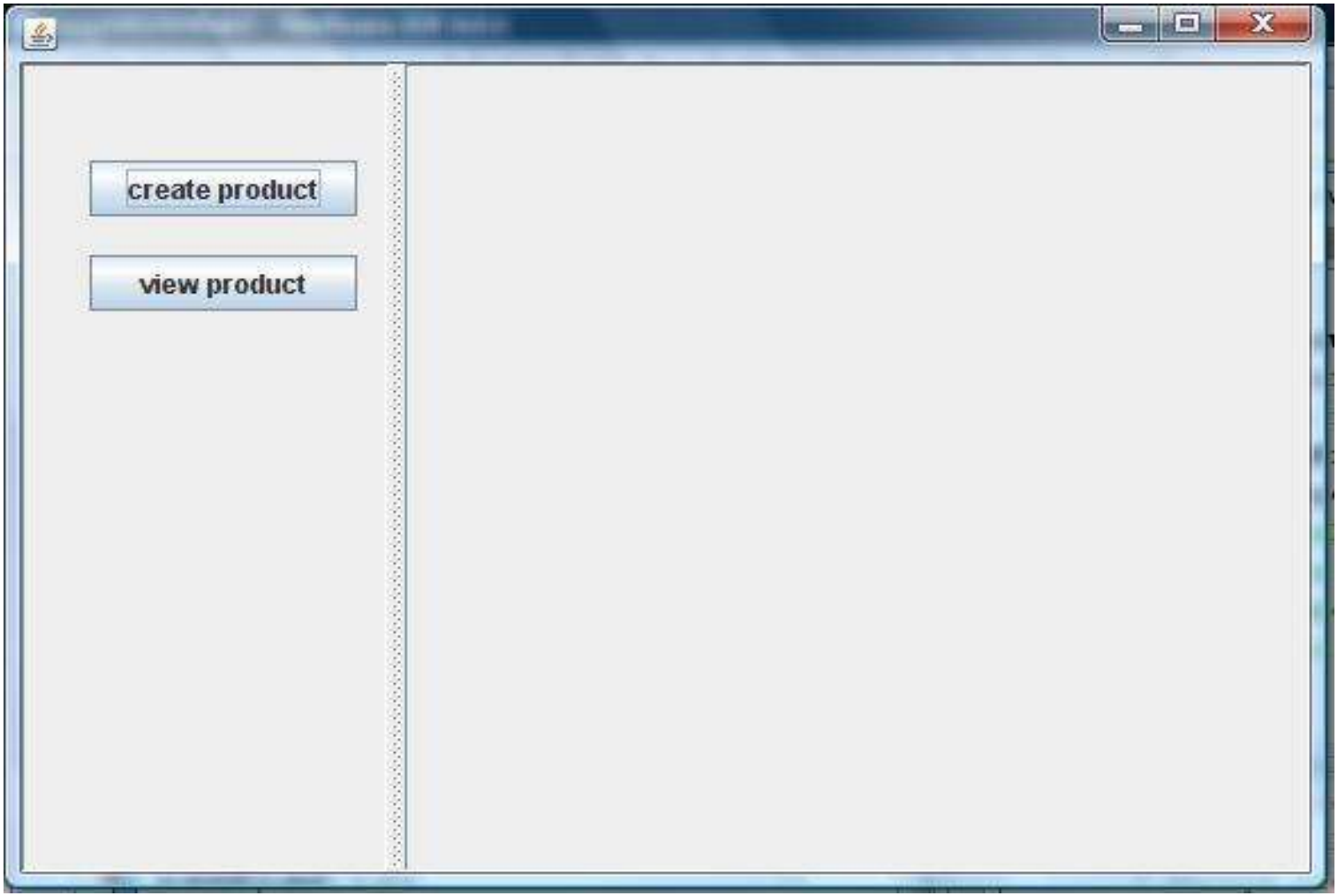
Person


```
class Person {

    lastName: String
    firstName: String
    address:    String

    getLastName();
    getFirstName();
    getFullName();
    getAddress();

    setLastName()
    setFirstName()
    setAddress();

    }
```

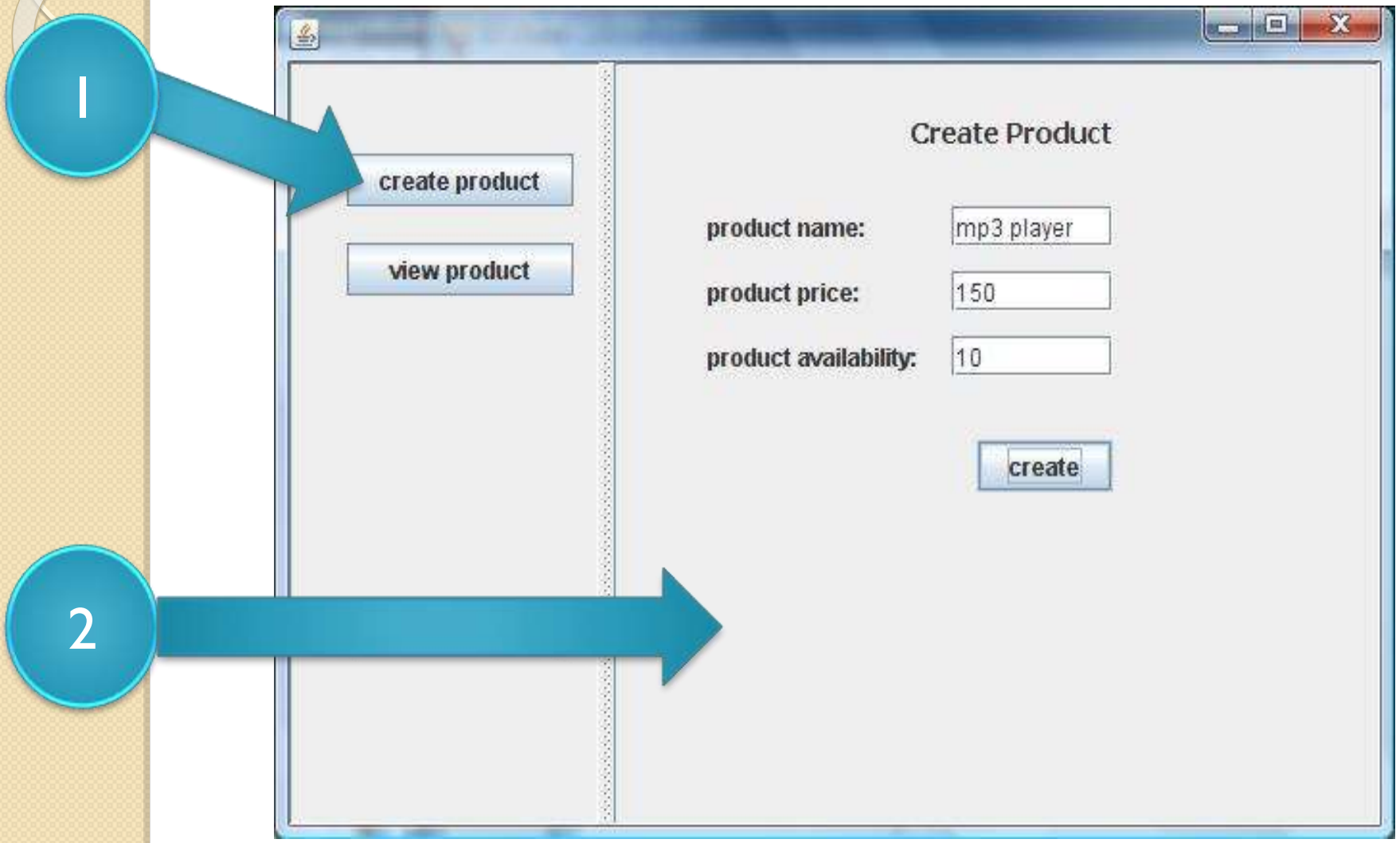



```
Person p = new Person();
String fn = firstnameTextField.getText();
p.setFirstName(fn);
….
```

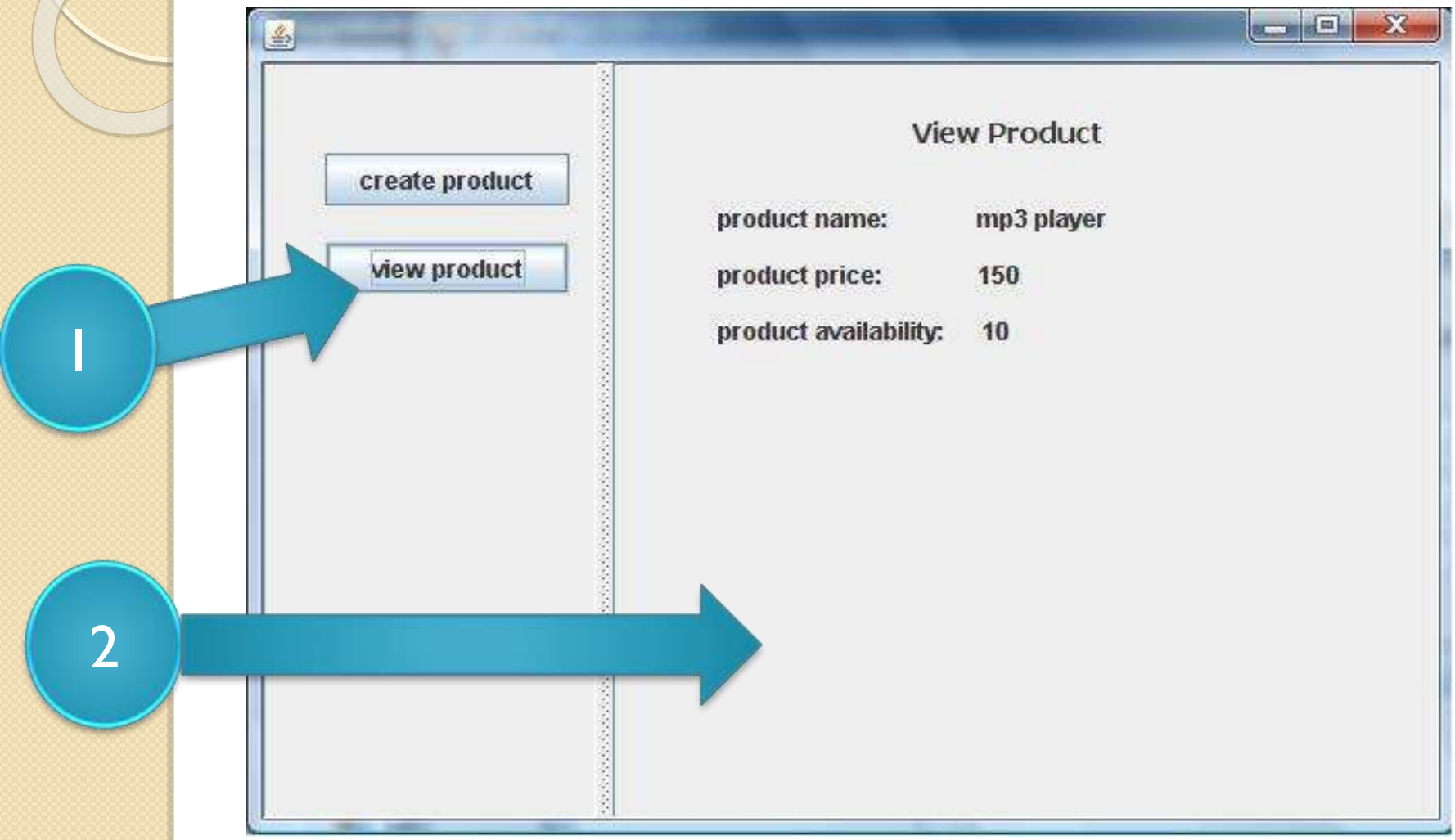
Output Application 1



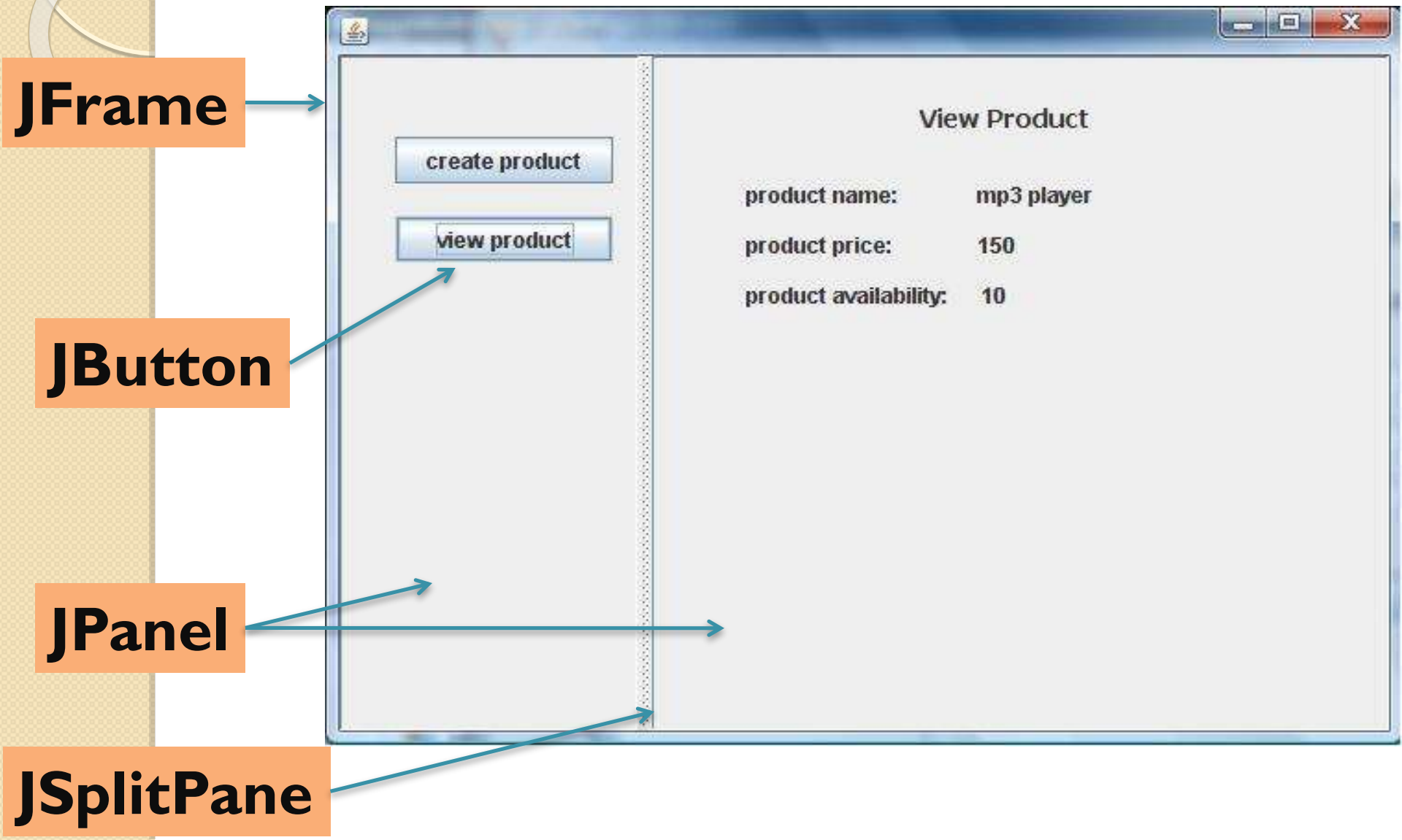
Output Step 1



Output Step 2



Swing application structure



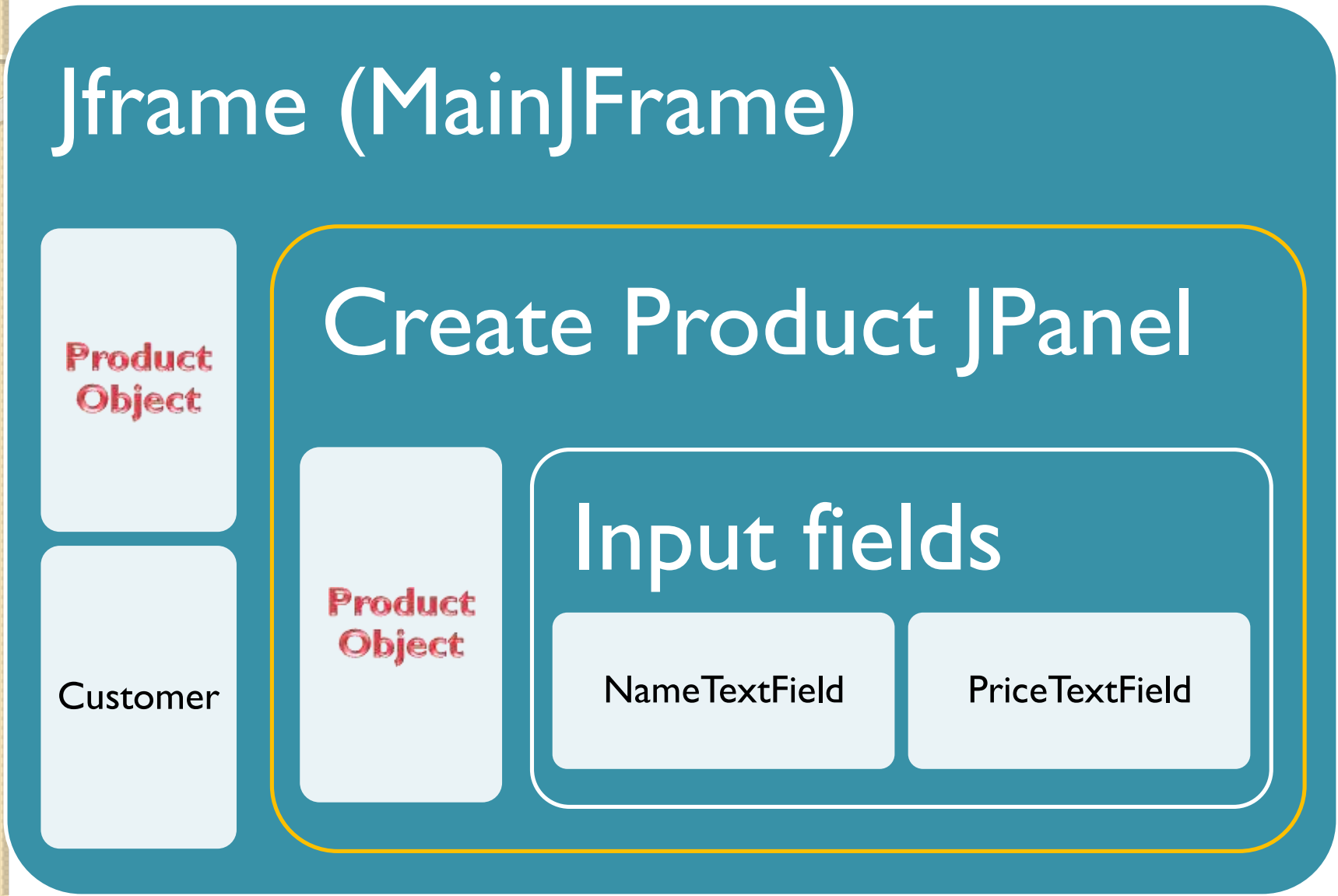
One UI and two tasks (create/View)

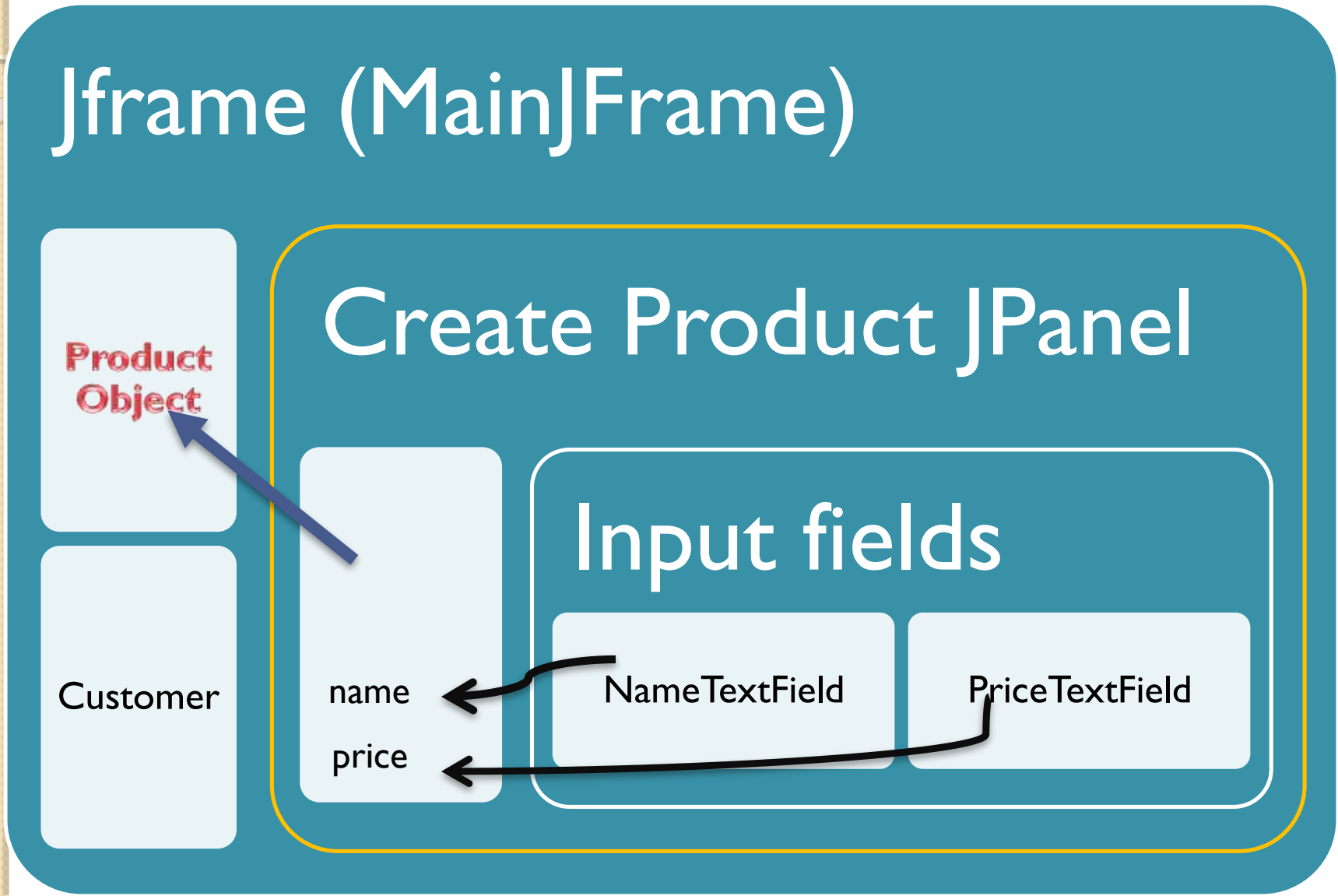
Jframe (MainJFrame)

Jpanel for create a product

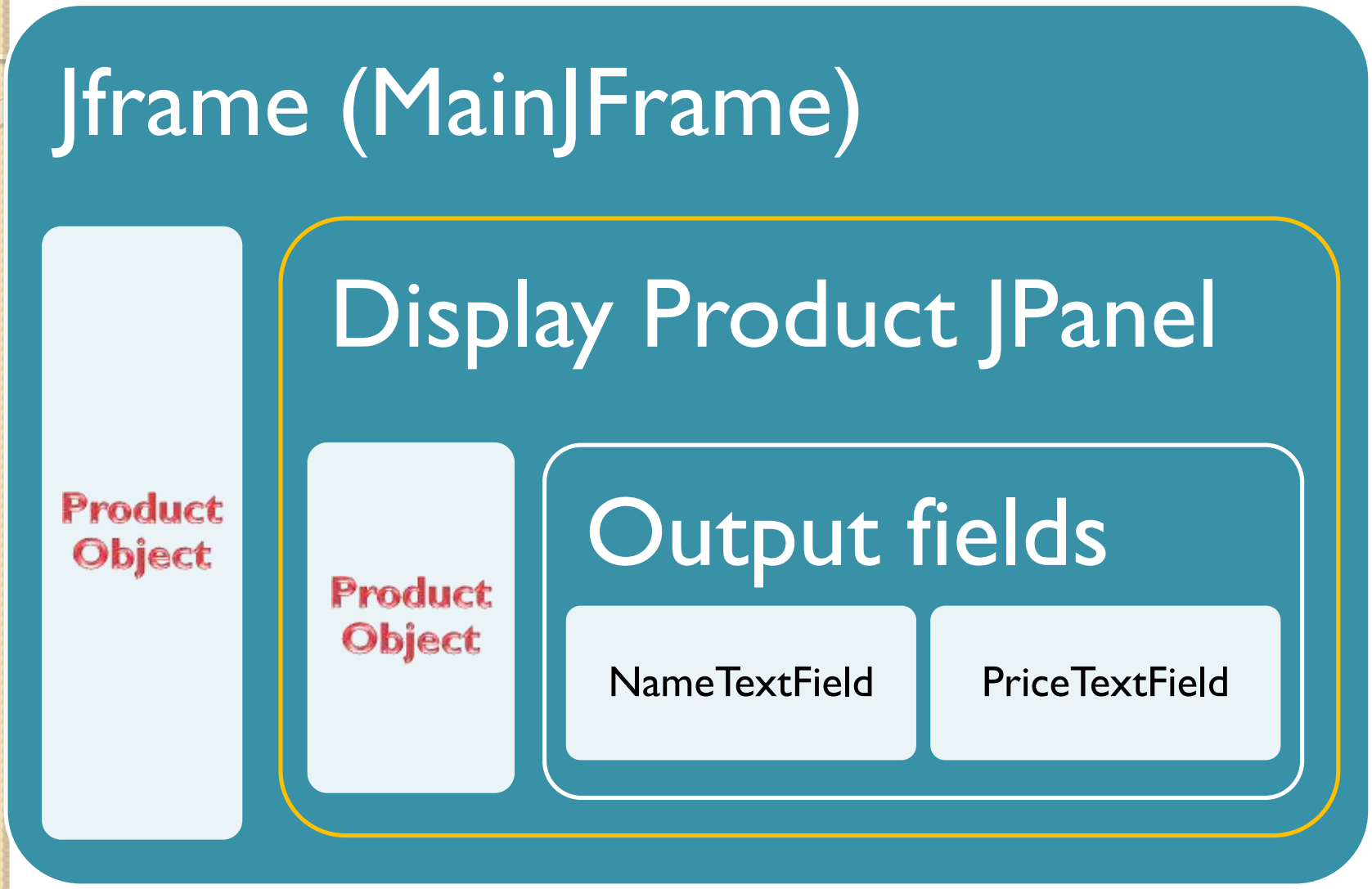
Jpanel for displaying a product

Pass the product object

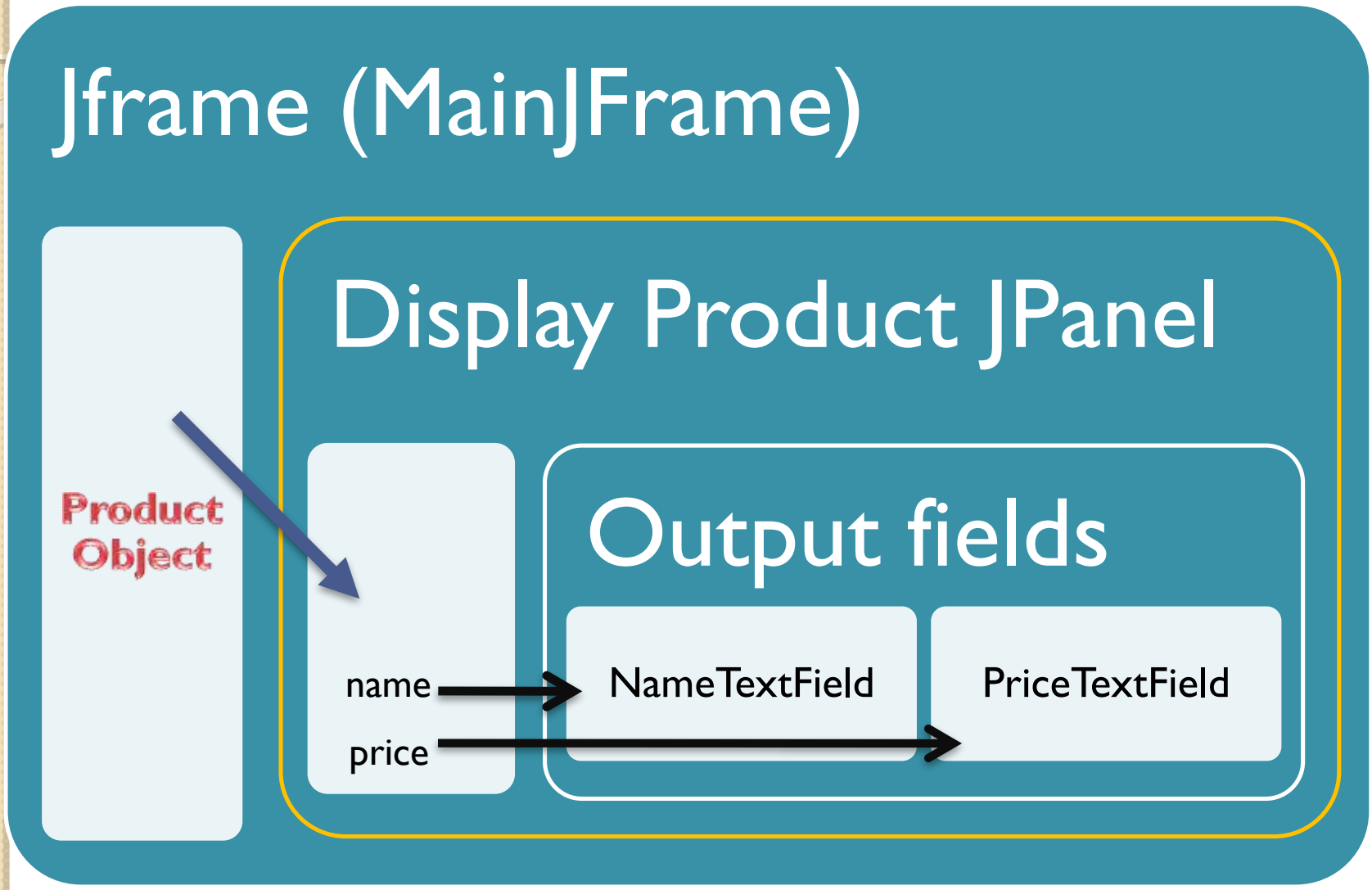



Jframe (MainJFrame)
Product Object
Customer
Create Product JPanel
name
price
Input fields
NameTextField
PriceTextField

Putting the display panels together



Putting the display panels together

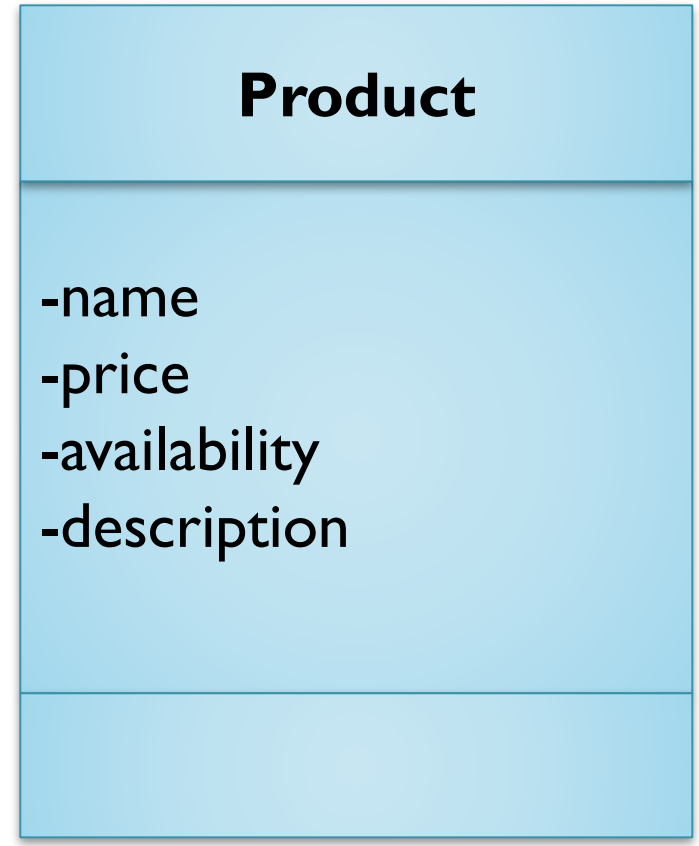


Procedure

1. Create a new project
2. Define Business Package
   - Create product class
3. Define UserInterface Package
   - Define JFrame
   - Define Jpanels (Create and View)
     - Button listeners

Create Product Class

under the business package



Product Class

```
public class Product {

    private String name;
    private String price;
            :

    public String getName()    // retrieve data
    {       return name;       }
                        :

     public void setName(String n)   // keep data
    {       name = n;           }
                        :
}
```

Define MainJFrame Class

under the userinterface package

Create a global variable of product for this class

```
class MainJFrame {

    public MainJFrame() {

        initComponents();

    }

}
```

Constructor

Method that creates visual components

Define MainJFrame Class

under the userinterface package

Create a global variable of product for this class

```
class MainJFrame {
private Product product; // Global Variable

 public MainJFrame() {

      initComponents();
      product = new Product(); // Instantiate the object
  (global variable)
}
```

Define CreaCreateProductJPanel class

under the userinterface package

```
class CreateProductJPanel() {

    CreateProductJPanel() {
    initComponents();
  }

<other stuff>

}
```

Constructor

Method that creates visual components

Define CreaCreateProductJPanel class

under the userinterface package

```
class CreateProductJPanel() {

        CreateProductJPanel(Product p) {
         initComponents();
         product = p;


   }

<other stuff>

}
```

Constructor

Create Product

- In the constructor of **CreateProductJPanel** class

```
public CreateProductJPanel(Product p) {
    initComponents();

}
```

View Product

- In the constructor of **ViewProductJPanel** class

```
public ViewProductJPanel(Product p) {
        initComponents();
    this.product = p;

    nameTextField.setText(product.getName(
 ));

    priceTextField.setText(product.getPric
 e());

}
```

Button Events

When Create or View buttons are clicked on the left side, the following actions should be performed respectively.

Create Button

```
CreateProductJPanel j = new
  CreateProductJPanel(product);
jSplitPane1.setRightComponent(j);
```

View Button

```
 ViewProductJPanel v = new
  ViewProductJPanel(product);
jSplitPane1.setRightComponent(v);
```

Create Button

When “create button” is clicked, following actions should be performed in the action perform method of the button

```
p.setName(nameField.getText()); //
p.setPrice(priceField.getText());
p.setAvailability(availabilityField.getText());
```

View Button

When “create button” is clicked, following actions should be performed in the action perform method of the button

```
nameTextField.setText(p.getName());
priceTextField.setText(product.getPrice());
```

Programming relationships

- Show how to implement relationships between classes
- Show how java works to connect and access objects
- Show how to traverse relationships

The Business Model

Person

First name,
Last name,
Social security number
DOB
Address Line 1,
Address Line 2,
City
Country
Zipcode

Etc

Example: One person multiple addresses

Joe,
Smith,
290-29-2974
2/2/1986
360 Huntington Ave
Snell Engineering
Boston
MA
USA
02115
Etc.…

Work

Joe,
Smith,
290-29-2974
2/2/1986
100 Main Street,
Natick
MA
USA
01760
Etc.…

Local

Joe,
Smith,
290-29-2974
2/2/1986
201 Best street
Cool-town
Shanghai
China

Etc …..

home

Multiple addresses means duplication of information and potential for errors

Joe,
Smith,
290-29-2974
2/21/1990
360 Huntington Ave
Snell Engineering
Boston
MA
USA
02115
Etc.…

Work

Joe,
Smith,
200-29-2974
2/2/1986
100 Main Street,
Natick
MA
01760
USA
Etc.…

Local

Joe,
Summer,
290-29-2974
2/2/1986
201 Best Street
Cool-town
Shanghai
China

Etc …..

home

What if we split the info into:

First name,
Last name,
Social security number
DOB

Address Line 1,
Address Line 2,
City
Country
Zipcode

What if we split the info into:

First name,
Last name,
Social security number
DOB
Address

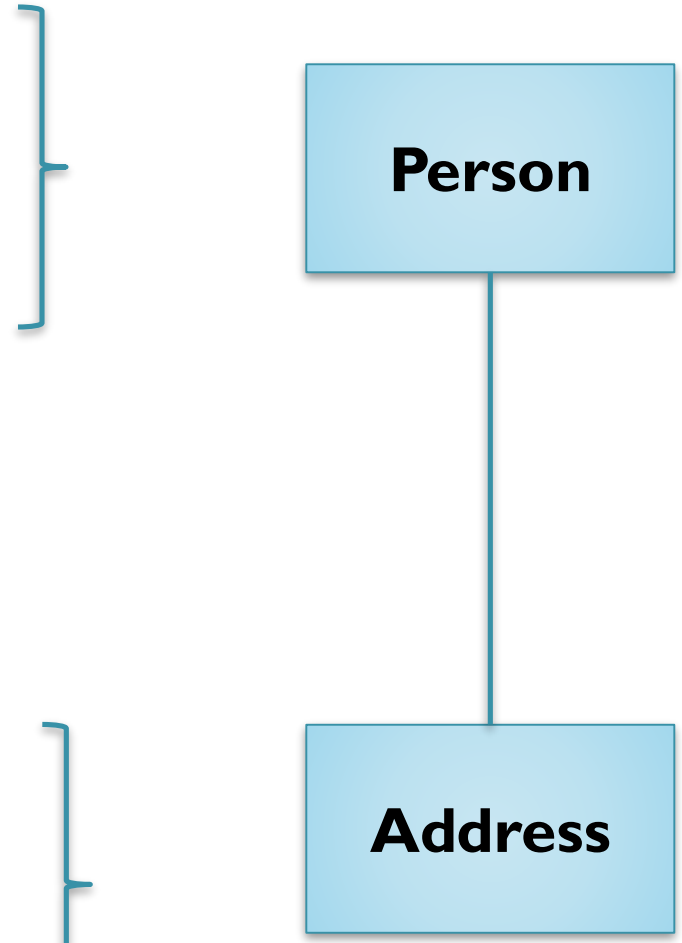
Personal information

Address Line 1,
Address Line 2,
City
Country
Zipcode

Address specific

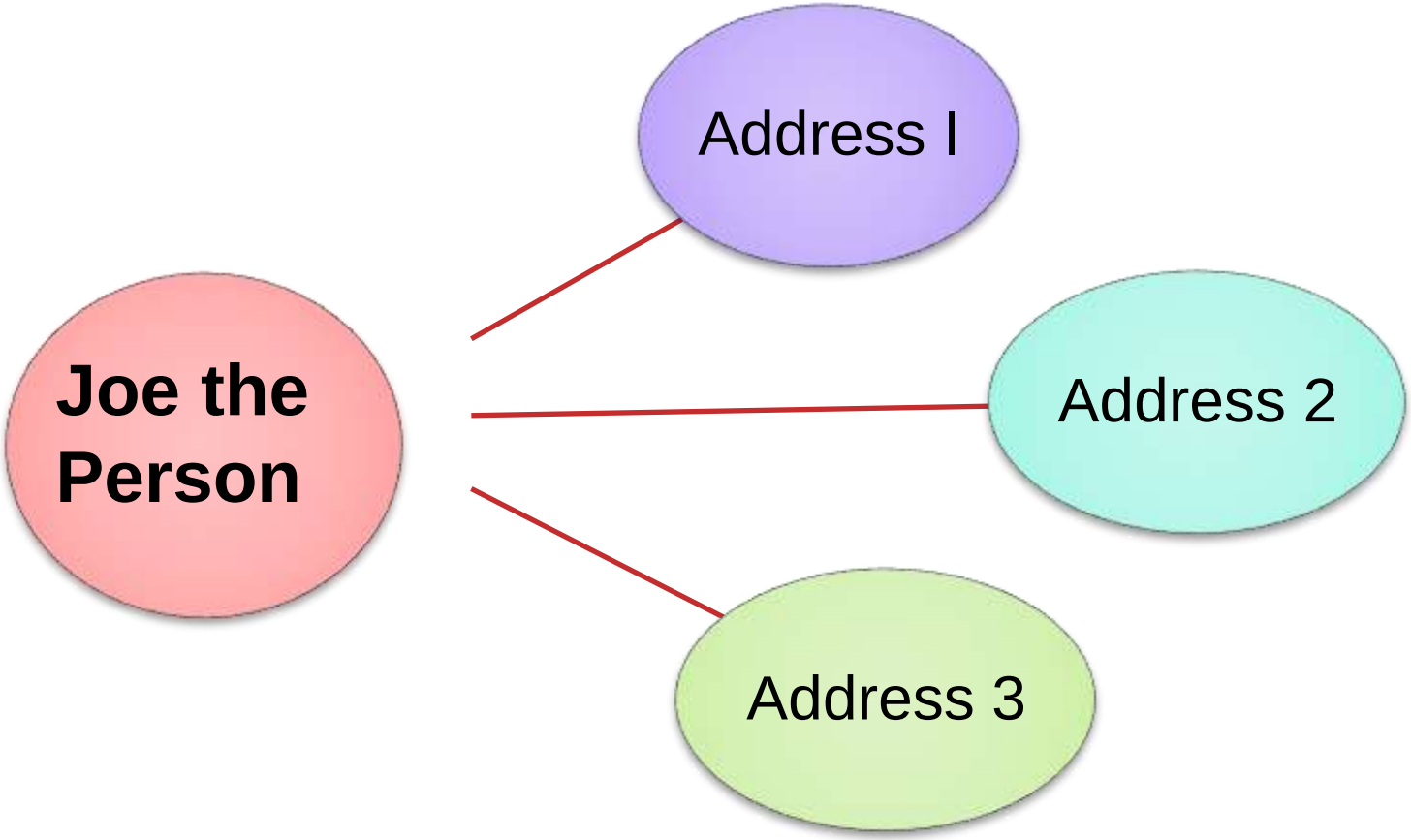
What if we split the info into:

First name,
Last name,
Social security number
DOB
Address

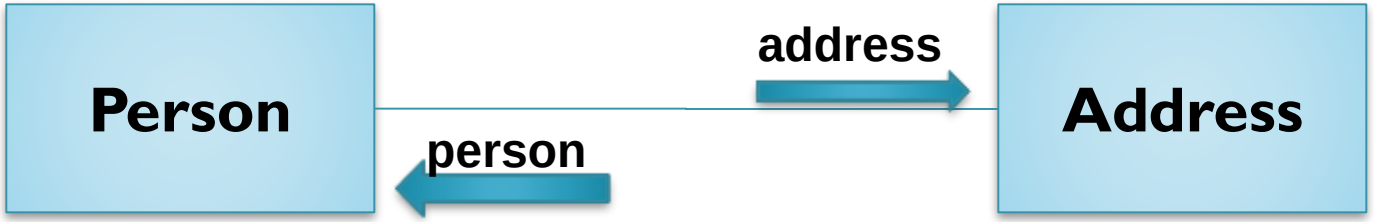


Address Line 1,
Address Line 2,
City
Country
Zipcode

One person object linked to three address objects each keeping track of one address



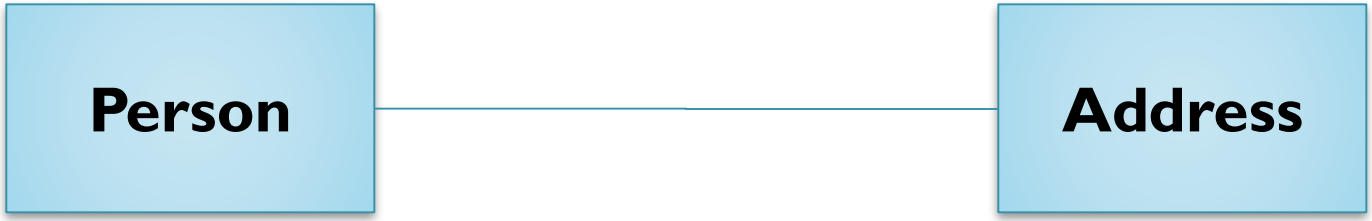
A reference variable is needed in the person class to keep track of the address object



From the person perspective, we must add

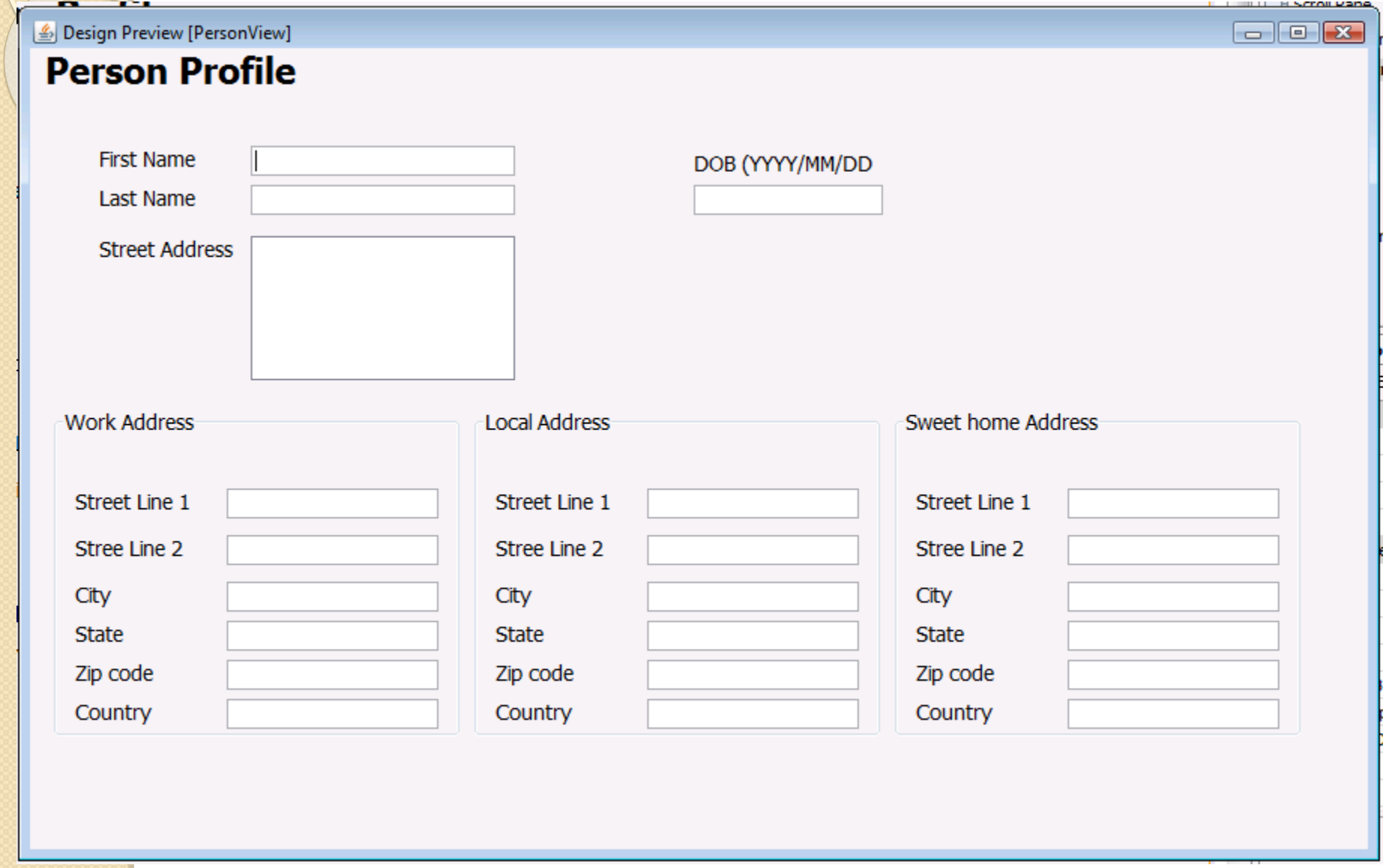
1) **An address attribute which will serve as a reference variable to the address object**
2) **getAddress method to retrieve the address object**
3) **setAddress method to store the address object as part of the person**

But how do we differentiate different kinds of addresses from each other such as work, home,…?



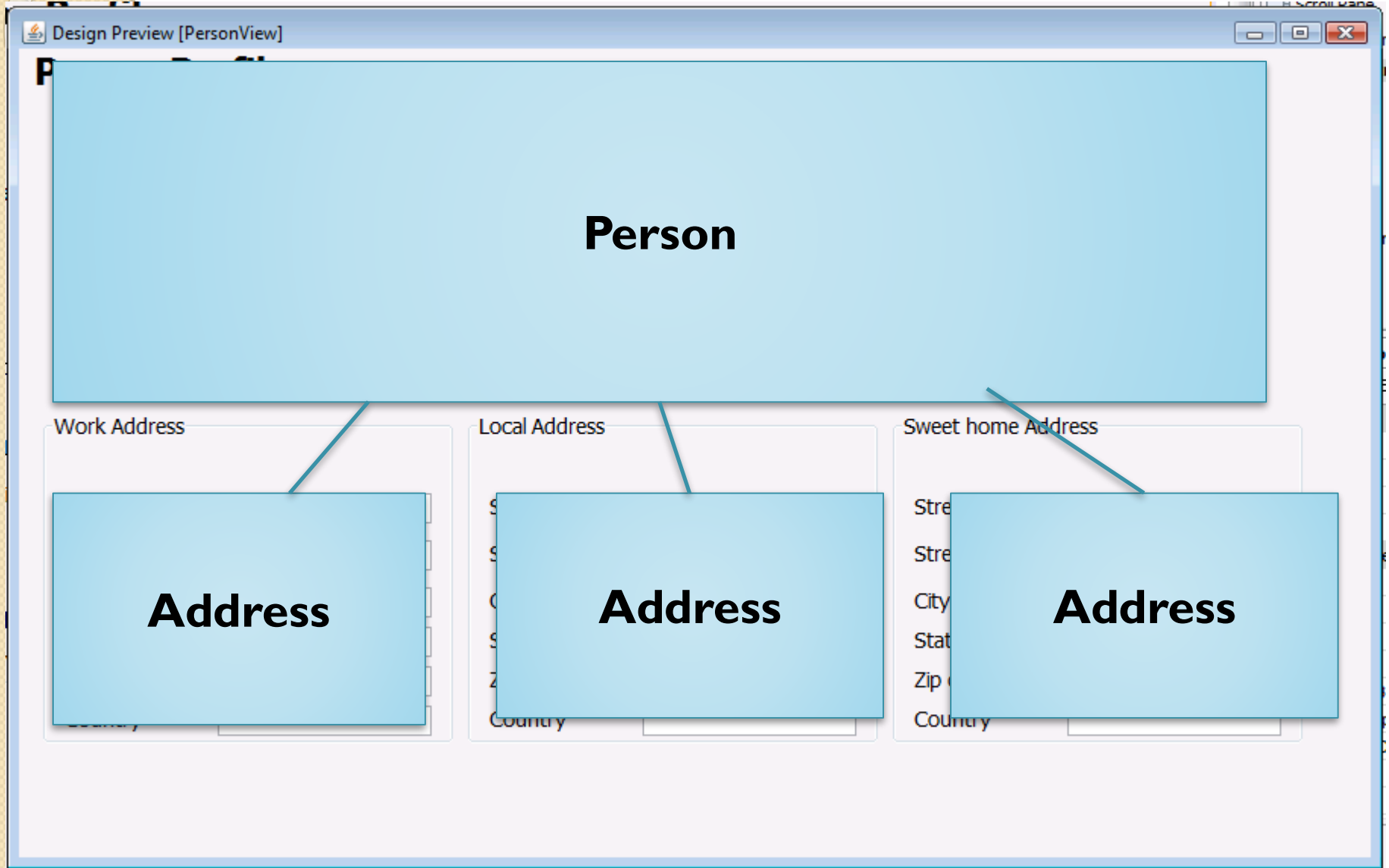
Answer: we can't with this kind of relationship unless we add an address type attribute to the address class

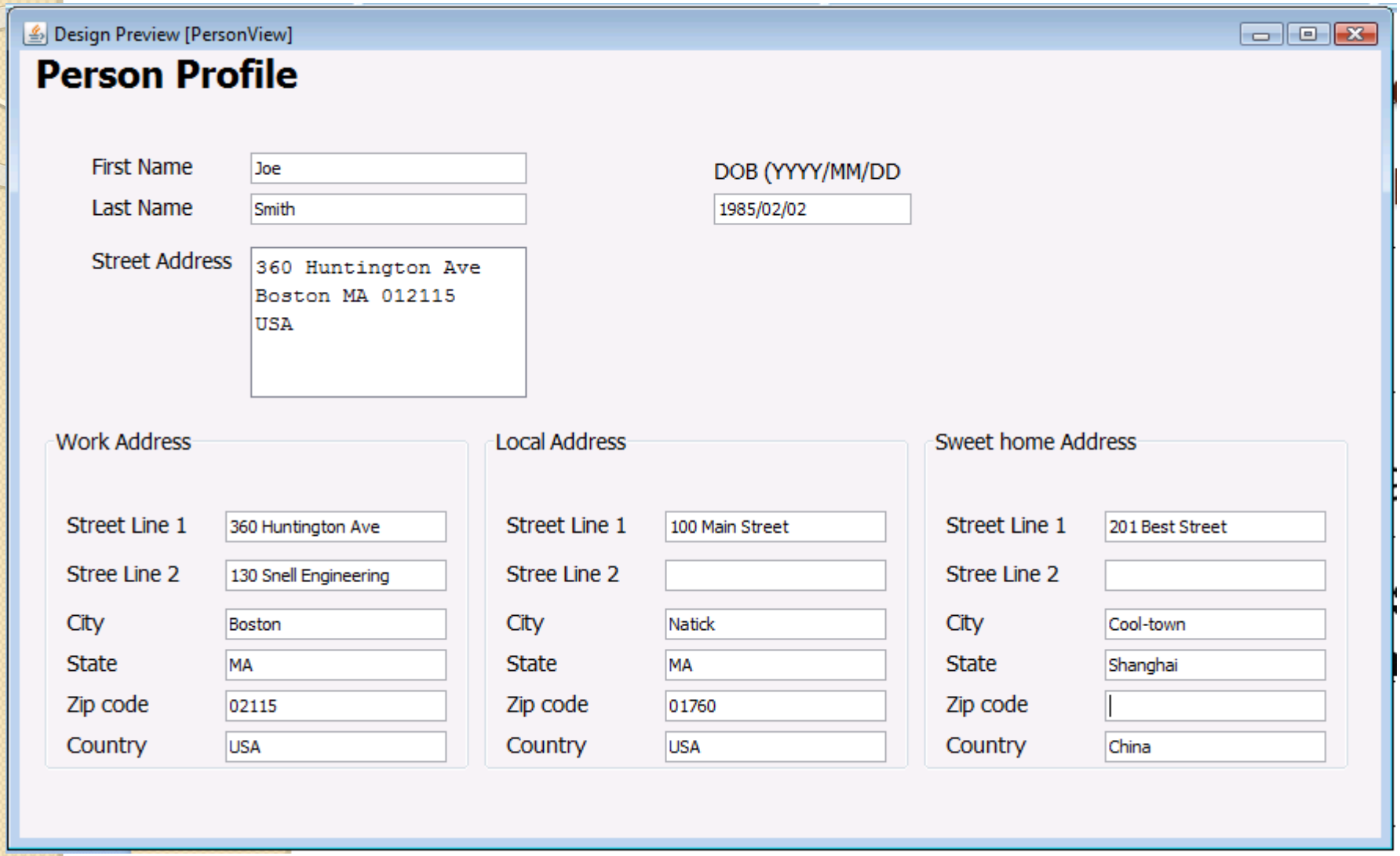
View Person Profile Screen



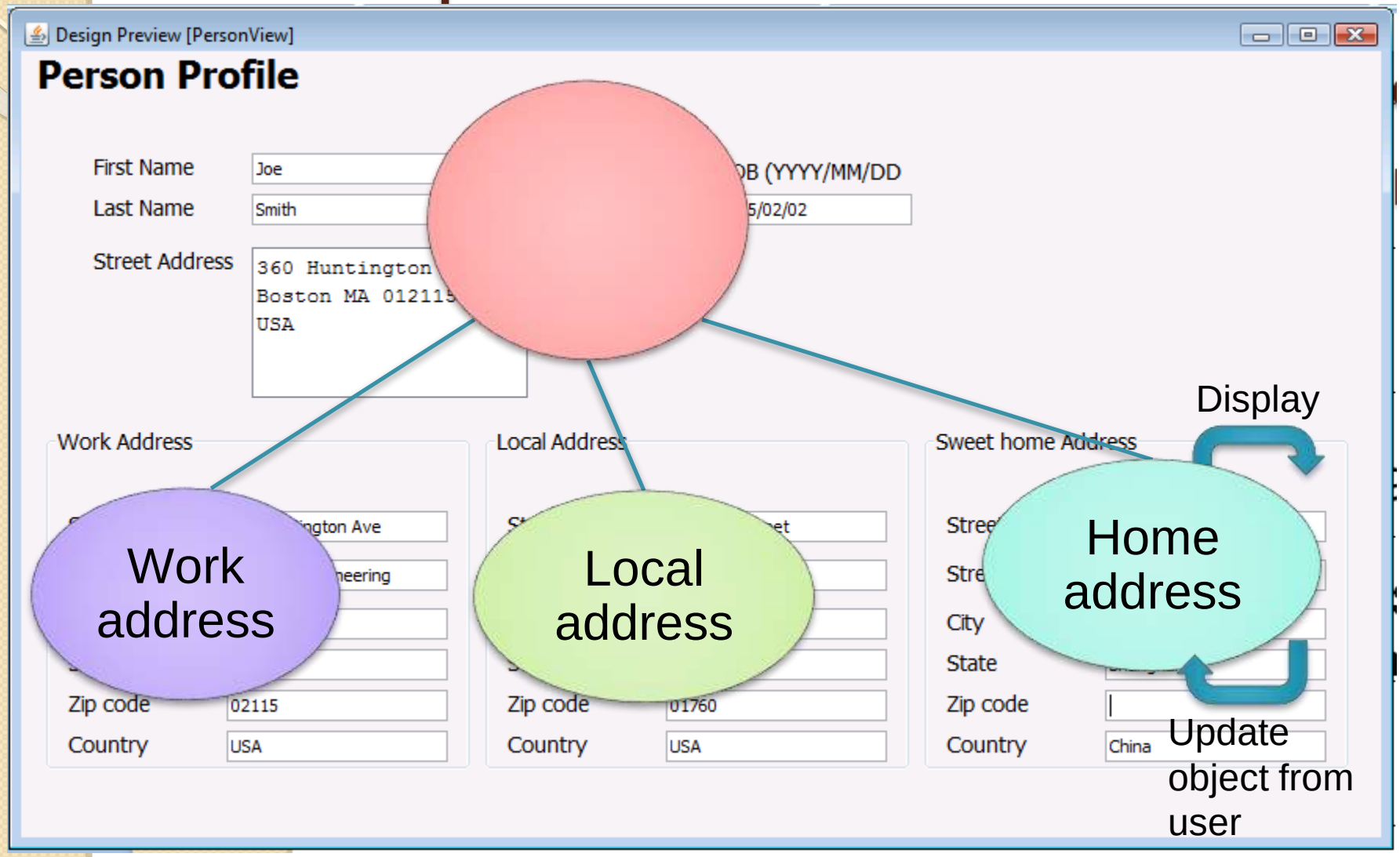

Design Preview [PersonView]
Person Profile
First Name
Joe
Last Name
Smith
DOB (YYYY/MM/DD
1985/02/02
Street Address
360 Huntington Ave
Boston MA 012115
USA
Work Address
Street Line 1
360 Huntington Ave
Stree Line 2
130 Snell Engineering
City
Boston
State
MA
Zip code
02115
Country
USA
Local Address
Street Line 1
100 Main Street
Stree Line 2
City
Natick
State
MA
Zip code
01760
Country
USA
Sweet home Address
Street Line 1
201 Best Street
Stree Line 2
City
Cool-town
State
Shanghai
Zip code
Country
China

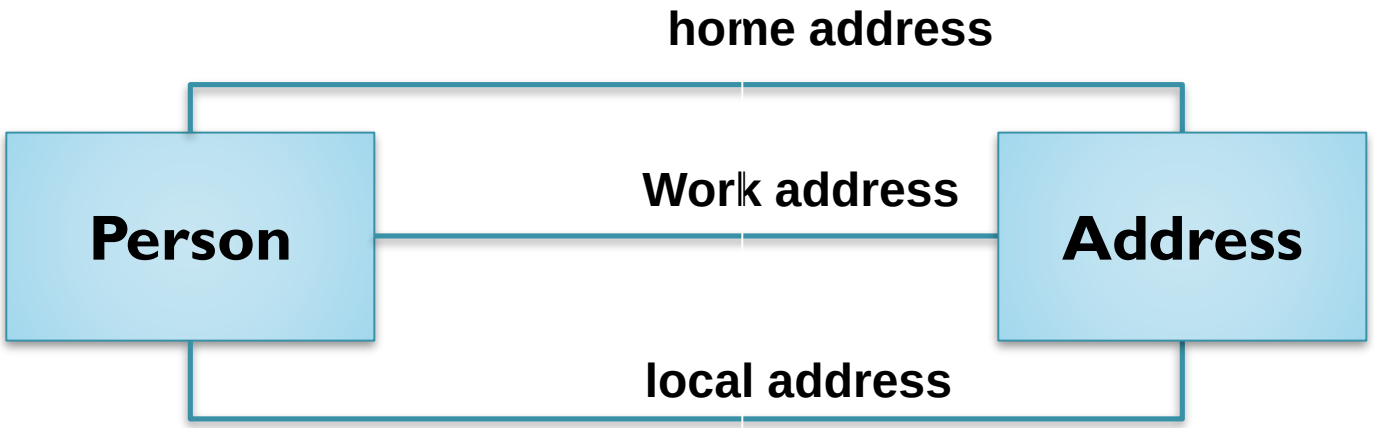
Matching the model to the user interface




Design Preview [PersonView]
Person Profile
First Name
Joe
Last Name
Smith
DOB (YYYY/MM/DD
1985/02/02
Street Address
360 Huntington Ave
Boston MA 012115
USA
Work Address
Street Line 1
360 Huntington Ave
Stree Line 2
130 Snell Engineering
City
Boston
State
MA
Zip code
02115
Country
USA
Local Address
Street Line 1
100 Main Street
Stree Line 2
City
Natick
State
MA
Zip code
01760
Country
USA
Sweet home Address
Street Line 1
201 Best Street
Stree Line 2
City
Cool-town
State
Shanghai
Zip code
Country
China

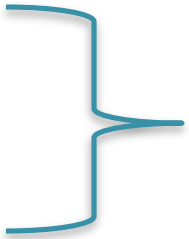
Changes are localized at the single address – person info remains fixed




home address
Person
Work address
Address
local address

```
Class Person {

String lastName;
String firstName;
Address workaddress;
Address homeaddress;
Address localaddress;

public Address getWorkAddress(){
Return workaddress;
}
Public void setWorkAddress(Address addressparam){
Workaddress = addressparam;
}
:::
}
```

```
Class Person {

String lastName;
String firstName;
Address workaddress;
Address homeaddress;
Address localaddress;

public Address getWorkAddress(){
Return workaddress;
}
Public void setWorkAddress(Address addressparam){
Workaddress = addressparam;
}
:::
}
```



3 reference variables for keeping track of the 3 addressess

Implementation of the class Person

```
Class Person {

String lastName;
String firstName;
Address workaddress;
Address homeaddress;
Address localaddress;

                                ddress(){


                                ss(Address addressparam){
                                aram;

:::
}
```

The reference variables hold addresses only. The must be declared of type Address

```
Class Person {

String lastName;
String firstName;
Address workaddress;
Address homeaddress;
Address localaddress;

public Address getWorkAddress(){
Return workaddress;
}
Public void setWorkAddress(Address addressparam){
Workaddress = addressparam;
}
:::
}
```

Implementation of the class Person

```
Class Person {

String lastName;
String firstName;
Address workaddress;
Address homeaddress;
Address localaddress;

public Address getWorkAddress(){
return workaddress;
}
:::
}
```

The get method returns the workaddress object.
The return value must be of type Address

Implementation of the class Person

Class Person {

The set method returns nothing (void). It takes an address object as an input parameter and saves it in the workaddress attribute. As a result the person would be linked to an address object as the work address.

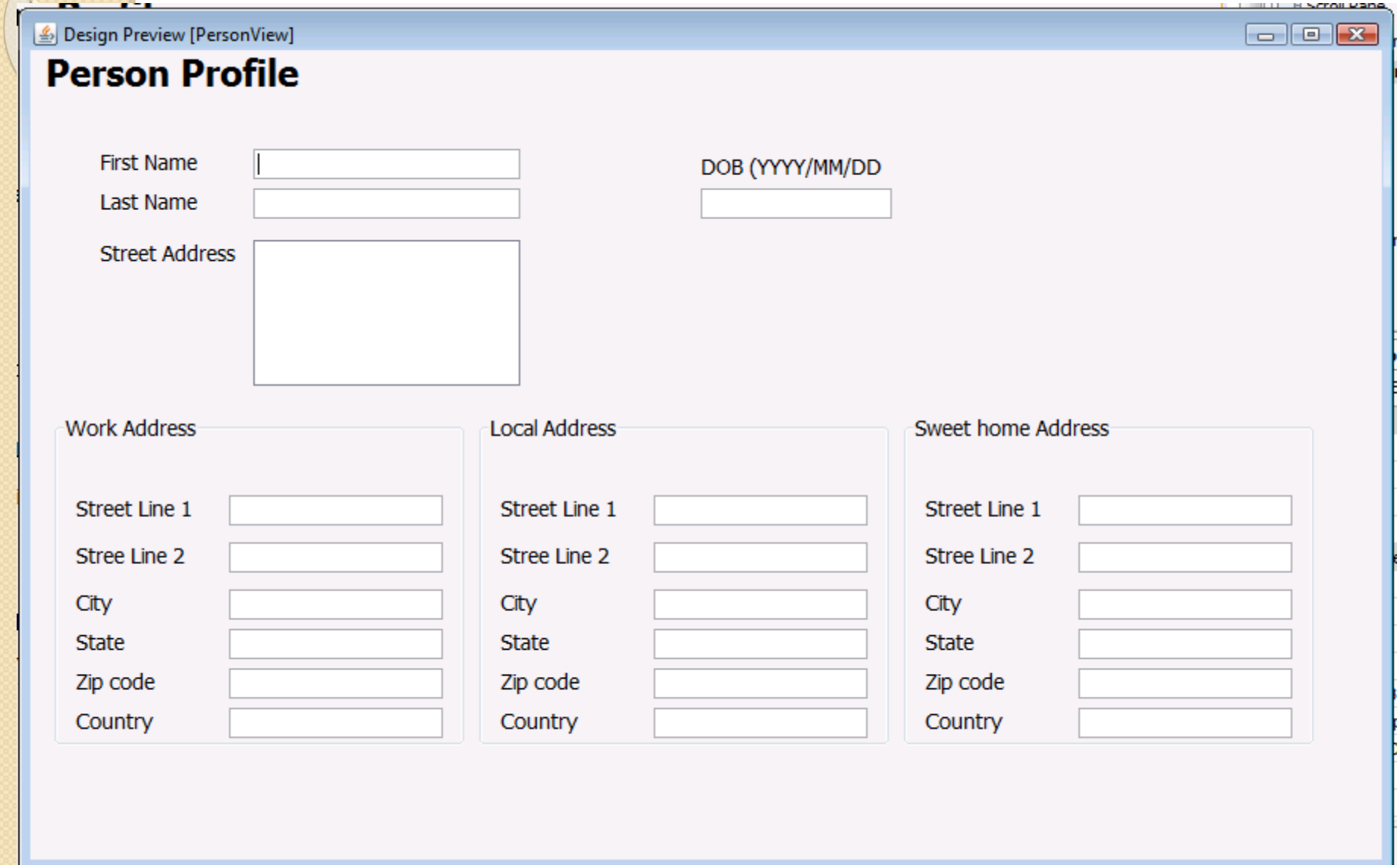
}

```
Public void setWorkAddress(Address addressparam){
workaddress = addressparam;
}
:::
}
```

Implementation of the class Person

```
Person  person = new Person();

person.setFirstName("Joe");
person.setLastName("Smith");

address address = new Address();
address.setAddressLine1("360 Huntington Ave");
person.setWorkAddress(address); //save address object as a work address

address = new Address(); // create new address object using the same ref variable
address.setAddressLine1("100 Main Street"); //insert address info
person.setLocalAddress(address);//save address object as a local address

address = new Address();
address.setAddressLine1("201 Best Street");
Person.setHomeAddress(address);
```

Consider the following form



Exercise 1

- Define a project with a main class.
- Define the classes as described
- In the main method do the following:

  Create a person object and three address objects. Initialize the objects with sample data.

  Use the system.printlin function to print all your data in a format like the following:

Person

1. First name : Joe
2. Last name: Smith

Work Address

1. Address Line 1: 360 Huntington Ave.
2. Etc….

Exercise 2

- Define a project with a jframe
- In the constructor for the jframe initialize a person object and three address like before.
- Pass the person object to a jpanel to display the person info as showing the form defined earlier.
- In the constructor for the jpanel display the person and 3 addresses to the jpanel form.