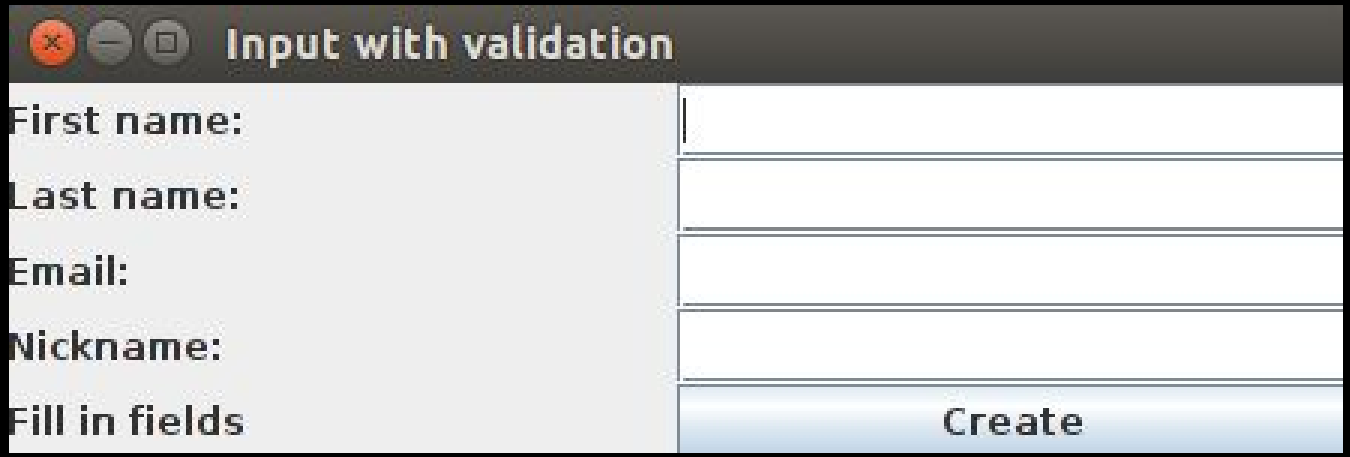
Input validation

Checking the user input

Kontroll av att alla fält är ifyllda

Skapa konto:



Om ett eller flera fält är tomma?

- Visa ett meddelande för varje fel?
- Visa ett meddelande med sammanfattning?

Skapa sammanfattande meddelande

```
StringBuffer errorMsg = new StringBuffer();
if(firstNameTextField.getText().equals("")){
  errorMsg.append("First name\n");
}
if(lastNameTextField.getText().equals("")){
  errorMsg.append("Last name\n");
}
//... and so on...
```

Hur vet vi att något saknades?

```
// Something like this:
if(errorMsg.length()!=0){
  // Show error message
}else{
  // Show OK and create the account
}
```

Var gör vi denna validering?

```
// In the listener for the createButton?:
public void actionPerformed(ActionEvent ae){
  StringBuffer errorMsg = new StringBuffer();
  if(firstNameTextField.getText().equals("")){
    errorMsg.append("First name\n");
  } // ... and so on
}
```

Separera ansvar, metoder och objekt

```
// Make check a separate method!
public void actionPerformed(ActionEvent ae){
  handleInputFields();
}
public void handleInputFields(){
  if <all fields are correctly filled in>{
    createAccount();
  }else{
    showErrorMessage();
  }
}
```

checkFields() - hur får vi feedback?

Hur vet vi om checkFields() gick bra?

- Vi kan använda Exceptions (halvbra)
  - Vi får skapa en egen Exception-klass
- Vi kan använda ett Validation-objekt (bättre)
  - Vi får skapa Validation-klassen själva

Strategi 1 - Exception

```
public void actionPerformed(ActionEvent ae){
  try{
    checkFields();
    resultLabel.setText("OK");
  }catch(InputValidationException ive){
    showErrorMessage(ive.getErrors());
    resultLabel.setText("Missing fields");
  }
}
```

checkFields() with Exception

```
private void checkFields() throws InputValidationException{
     InputValidationException ive = new InputValidationException();
     if(!isValidInput(firstNameTextField.getText())){
         ive.addError("First name");
     }
     if(!isValidInput(lastNameTextField.getText())){
         ive.addError("Last name");
     }
     if(!isValidInput(emailTextField.getText())){
         ive.addError("Email");
     }
     if(!isValidInput(nicknameTextField.getText())){
         ive.addError("Nickname");
     }
     if(ive.hasErrors()){
         throw ive;
     }
 }
```

InputValidationException

```
public class InputValidationException extends Exception{
    // We use a List (interface type) so that we may change
    // from ArrayList to some other List later if we need to
    private List<String> errors = new ArrayList<String>();
    public void addError(String msg){
        errors.add(msg);
    }

    public List<String> getErrors(){
        return errors;
    }

    // If the list isn't empty, there are errors!
    public boolean hasErrors(){ return !errors.isEmpty(); }
}
```

Strategi 2 - Ett Validation-objekt

```
public void actionPerformed(ActionEvent ae){
  Validation validation = new Validation();
  checkFields(validation);
  if(validation.passesValidation()){
    resultLabel.setText("OK");
  }else{
    showErrorMessage(validation.getErrors());
    resultLabel.setText("Missing fields");
  }
}
```

checkFields() with Validation object

```
private void checkFields(Validation validation){
  if(!isValidInput(firstNameTextField.getText())){
    validation.addError("First name");
  }
  if(!isValidInput(lastNameTextField.getText())){
    validation.addError("Last name");
  }
  if(!isValidInput(emailTextField.getText())){
    validation.addError("Email");
  }
  if(!isValidInput(nicknameTextField.getText())){
    validation.addError("Nickname");
  }
}
```

Validation-klassen

```
public class Validation{
    List<String> errors = new ArrayList<String>();
    public void addError(String error){
        errors.add(error);
    }
    public boolean hasErrors(){
        return ! errors.isEmpty();
    }
    public boolean passesValidation(){
        return !hasErrors();
    }
    public List<String> getErrors(){
        return errors;
    }
}
```

Skillnad i flödet/logiken

```
// Using exceptions
public void actionPerformed(ActionEvent ae){
  try{
    checkFields(); // ←Here, an exception might occur
    resultLabel.setText("OK");
  }catch(InputValidationException ive){
    showErrorMessage(ive.getErrors()); //←We might go here
    resultLabel.setText("Missing fields");
  }
}
// Not very clear flow of execution
// But easy to read what should happen, when all is fine
```

Flödet i Validation-fallet

```
public void actionPerformed(ActionEvent ae){
  Validation validation = new Validation();
  checkFields(validation);
  if(validation.passesValidation()){
    resultLabel.setText("OK");
  }else{
    showErrorMessage(validation.getErrors());
    resultLabel.setText("Missing fields");
  }
}
// Straight forward:
// * Create
// * Check validation
// * Make decision depending on how validation went
```