1. Criação de um projeto (ver item 4)

- Numa interface de linha de comando aberta na pasta de projetos da disciplina, digite:

```
lein new app <nome>
```

- Edite o arquivo `project.clj` para incluir bibliotecas como dependências do projeto se for o caso. Em seguida, entre na pasta do projeto e digite:

```
lein deps
```

- Inicie o IntelliJ IDEA. Na tela inicial, selecione *Import Project*.
- Encontre a pasta principal do projeto e siga os passos, as únicas escolhas relevantes são o Leiningen e o JDK Oracle.

Configuração do REPL (normalmente realizada apenas uma vez)

- No IntelliJ IDEA, acione o menu *Run→Edit Configurations...*
- Na caixa de diálogo que se abre, selecione + → *Clojure REPL* → *Local*
- Em *name*, digite: *REPL user*
- Escolha a opção "*Use nREPL in normal JVM process*"
- Desabilite os itens existentes em "*Before launch*" selecionando-os, e clicando em – em seguida.
- Acione o REPL em *Run* → *Run REPL user*, ou através da *combobox* e do ícone *play* verde na barra de ferramentas.
- Configure o REPL para aparecer na parte de baixo (*Bottom*) da IDE.

2. Criação e edição de um *namespace* (arquivo de programa)

- Inicie o IntelliJ IDEA e escolha o projeto já existente. Se ele não aparecer na lateral esquerda (últimos projetos abertos), acione a opção *Open* e encontre a pasta principal do projeto.
- No painel *Project*, abra a pasta `src/<nome>`
- Com o botão contrário do mouse, acione o menu suspenso e selecione *New* → *Clojure Namespace* e dê um nome ao *namespace*, que também será o nome do arquivo, seguido da extensão `.clj`
- No arquivo, configure a macro `ns` inicial de forma a carregar as bibliotecas e funções desejadas. Por exemplo, para carregar as funções `split-lines` e `join` do *namespace* `clojure.string`:

```
(ns <nome-do-namespace>
  [:require [clojure.string :refer [split-lines join]]])
```

- Edite o arquivo (ver item 5).
- Para fazer funcionar o programa, há duas opções:
  - Carregue-o num REPL e experimente as funções (ver item 3).
  - Simplesmente execute o programa através do menu *Run* → *Run...* seguido da escolha do *namespace*.[1]

3. Iniciando uma sessão de REPL[2]

- Acione *Run* → *REPL user*, ou a *combobox* e o ícone *play* verde.
- O REPL é iniciado no *namespace* `user`. Somente nesse *namespace* funcionam `doc`, `find-doc`, `source`.
- Clique dentro do arquivo (*namespace*) com o programa a ser testado. No menu *Tools* → *REPL* acione *Load File in REPL* (existe um atalho de teclado também)[3].

[1] A *combobox* na barra de ferramentas, que à direita tem o ícone acionável na forma de *play* e da cor verde, mantém como opção ativa a última escolha de execução de *Run...* , seja REPL ou *namespace* executado diretamente. A *combobox* e o ícone *play* verde são o caminho imediato para a configuração mais recente.

[2] Um REPL já iniciado pode não estar visível, pois o painel foi escondido. Traga-o em *View* → *Tool Windows*, ou use os acessos a painéis recolhidos que ficam nas margens estreitas da janela do IntelliJ IDEA.

[3] Durante a edição do programa, pequenas modificações podem ser recarregadas no REPL na base de funções individuais (*forms* em geral, na verdade) através do menu *Tools* → *REPL*, opções *Send top form to REPL* (há atalho) e *Send form before caret to REPL* (crie atalho em *File* → *Settings* → *Keymap*; invente nova combinação).

- No REPL (ainda no *namespace* `user`), é possível consultar a documentação das bibliotecas importadas com a macro `ns`. Use o nome qualificado (completo: *namespace* + função).

```
(doc quil.core/rect)
```

- Para entrar no *namespace* criado por você, digite

```
(in-ns '<nome-do-namespace>)
```

- Use os símbolos (funções e constantes) do seu programa sem incluir o nome do *namespace* na frente dos nomes dos símbolos.

4. O projeto `lab`

Inicialmente, crie um projeto `lab`, seguindo os passos no item 1. Ao longo da disciplina, vários pequenos programas são desenvolvidos nele, instanciados como novos *namespaces*. Com isso, minimiza-se a criação de projetos e ganha-se tempo.

Novos projetos devem ser criados para trabalhos; o procedimento no item 1 não deve ser corriqueiro.

5. *Structural Editing* do Cursive

O *Structural Editing* do Cursive tem origem no modo `paredit` (*par* de 'parênteses') do Emacs, criado para programação em Emacs Lisp (interpretador sobre o qual funciona o Emacs) e adotado também pela comunidade do Common Lisp, especialmente para o soberano ambiente de desenvolvimento para Common Lisp no Emacs, o Slime. Existem similares nos principais editores de texto para programação.

Como a programação em Lisp usa muitos parênteses, há grandes chances de o programador se equivocar na edição. O *structural editing* bloqueia a deleção de parênteses na forma tradicional. Os parênteses se tornam imunes ao *Delete* ou *Backspace*, exceto quando não contêm nada entre si. Isso pode causar estranheza. No entanto, há comandos (por atalhos principalmente) interessantes para manipulação de parênteses, muito diferentes daquilo com que se está acostumado em editores em geral.

Há quem ame e quem odeie o *structural editing.* Vale a pena insistir no início. Depois que se pega o jeito, compensa. Ele proporciona fluidez à edição dos programas. Altamente recomendado.

Observe as GIF animadas no site do Cursive e tente reproduzir os exemplos.

https://cursiveclojure.com/userguide/paredit.html

O *structural editing* deve estar habilitado (`Structural:On` na barra de status). Acesse os comandos do *structural editing* no menu *Edit* → *Structural Editing*. Aprenda a usar os atalhos indicados ali.