LAB3 – Laboratório 2 de Inteligência Artificial
Assunto: Algoritmo Minimax

I - Observações:

1. **O trabalho é em Dupla:** É permitido discutir com outras duplas os problemas e as estratégias para solucioná-los, mas as implementações das soluções, bem como as resoluções dos exercícios propostos devem ser feitas somente pelos componentes da dupla. Trabalhos plagiados da Internet ou de outras duplas não serão corrigidos e receberão nota 0 (zero).

2. **Forma de Entrega:** Entrega Eletrônica via **SINEF**. O relatório deve ser entregue em formato PDF (outros formatos não serão corrigidos) cujo nome deve ser: ***PrimeiroNome_UltimoNome_LAB3.pdf*** onde ***PrimeiroNome*** e ***UltimoNome*** são o primeiro e o último nome de algum dos membros da dupla. **Qualquer entrega fora deste padrão será penalizada em 20% da nota.**

3. **Data Limite para Entrega: 02/06/2010 (Quarta-Feira) até as 23:59 pelo SINEF**

II - O que deve ser entregue:

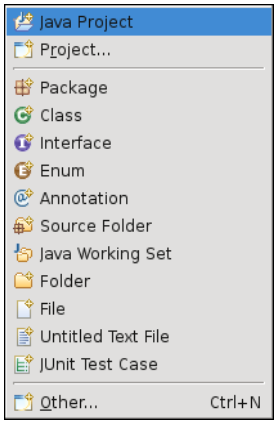
1. **Um relatório contendo:**
   a) os nomes dos componentes da dupla;
   b) os resultados obtidos com a execução dos programas das tarefas propostas;
   c) análise dos resultados obtidos.

2. **ATENÇÃO:** Não é preciso descrever a execução do tutorial (item III). Apenas deve ser entregue a resolução das tarefas propostas (item IV).

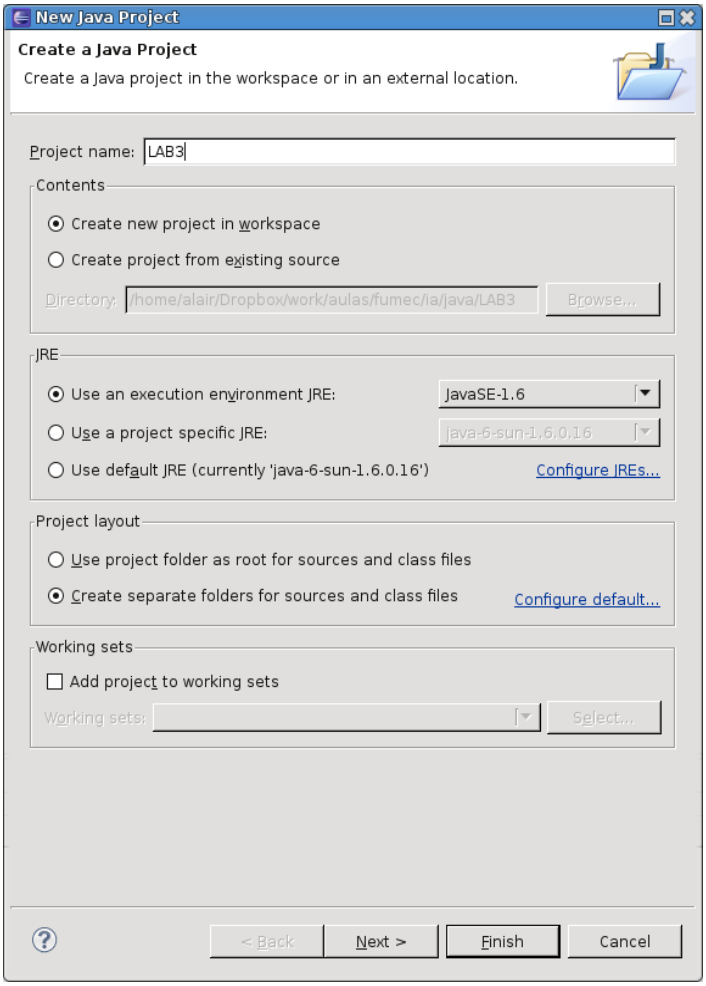
III – Tutorial Eclipse + Biblioteca da disciplina

Faça o download da biblioteca da disciplina e dos códigos fonte necessários, disponibilizada no SINEF com o nome ***recursos_lab3.zip***. Descompacte o arquivo dentro de uma pasta de sua escolha. A partir deste ponto, a pasta onde você descompactou o arquivo passará a ser chamada de *PASTA_LAB3*. Dentro da pasta *PASTA_LAB3*, depois de descompactado o arquivo zip, haverá um arquivo jar (*ia_tictactoe.jar*), uma pasta de nome *ia*, e uma pasta *docs*.

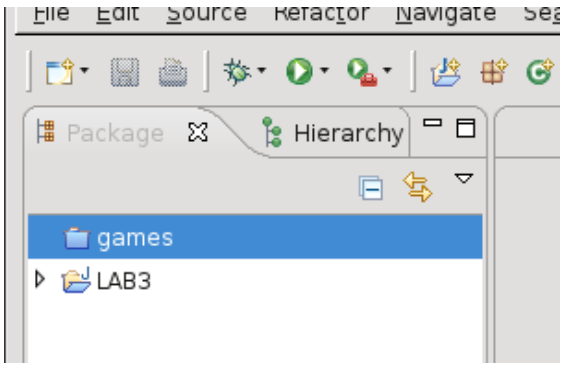
Abra o programa Eclipse. A versão do eclipse usada para confecção deste tutorial foi a 3.5.2. Caso a versão utilizada para execução seja diferente, as telas podem ser ligeiramente diferentes. No Eclipse, vá no menu *File → New e escolha Java Project*.



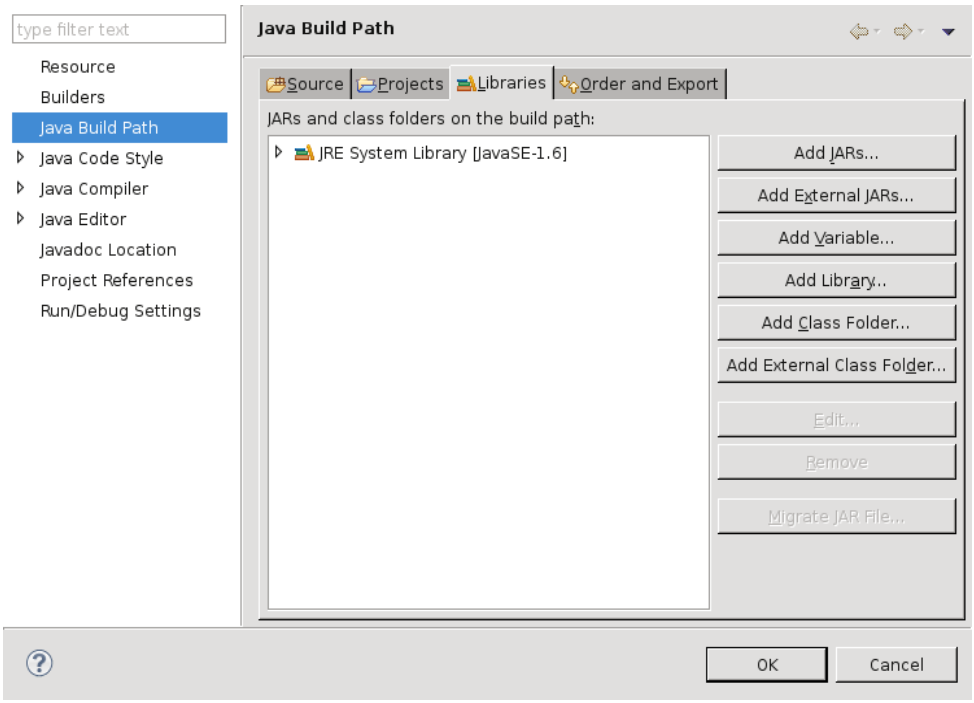
Uma janela como a da figura abaixo, aparecerá. Escreva o nome do seu projeto. Neste exemplo, utilizou-se o nome LAB3. As demais opções não precisam ser alteradas. Clique em *Finish*.



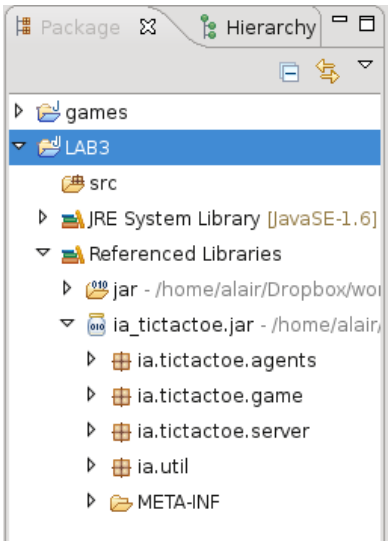
Uma pasta com o nome do projeto criado aparecerá do lado esquerdo, na aba *Package*, como mostrado na figura abaixo. Nesta figura, um outro projeto está disponível para trabalho (*games*). Qualquer projeto diferente do criado nesta etapa pode ser ignorado.



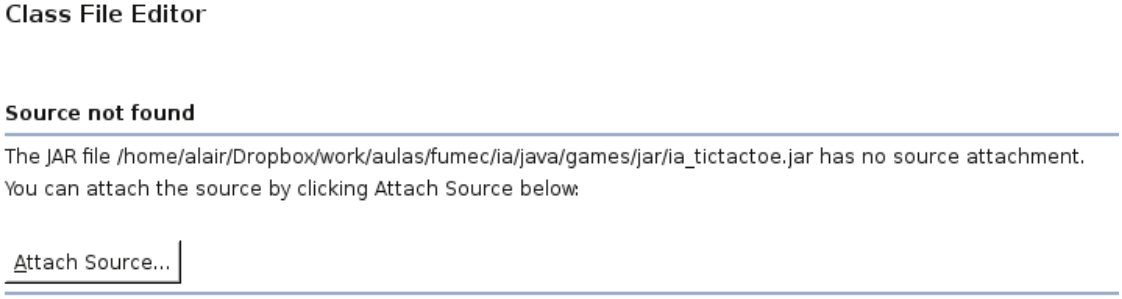
Clique com o botão direito na pasta do projeto (LAB3) e escolha a opção *Properties*. Na janela que se abrir, vá em *Java Build Path*, e selecione a aba *Libraries*, como mostrado na figura a seguir. Clique em *Add External JARs* e selecione o arquivo jar descompactado (*ia_tictactoe.jar*).



Expandindo-se a pasta do projeto *LAB3*, obtém-se uma árvore como a da imagem abaixo:



Se você tentar abrir alguma classe para ver o código fonte, não será possível, pois não existe nenhum arquivo fonte disponível. Uma tela como a seguinte aparecerá.



Clique no botão *Attach Source*, escolha a opção *External Folder* e selecione a pasta *PASTA_LAB3*, onde você descompactou o arquivo *zip*. Depois de fazer isto, você poderá ver o código fonte das classes *AgentKeyboard.class*, *BasicAgent.class* e *BitHead.class*. As demais classes não terão o código fonte disponibilizado neste momento. Neste laboratório você utilizará o código fonte da classe *BitHead*.

IV – Tarefas

1. Execute o programa *TicTacToeServer*. Para isso, clique com o botão direito no projeto que você criou e escolha a opção *Run As → Java Application*. O programa executará no console do Eclipse. Você deve escolher os agentes que jogarão o jogo. Existem dois agentes disponíveis:
2.
   a) ***AgentKeyboard***: Um agente que captura os comandos do teclado para jogar o jogo da velha (Tic-tac-toe);
   b) ***BitHead***: Um agente que implementa o algoritmo Minimax para o jogo da velha.

   Execute o jogo algumas vezes, alternando os jogadores entre você (*AgentKeyboard*) e o *BitHead*. Deixe que o agente *BitHead* jogue contra ele mesmo algumas vezes. Você conseguiu ganhar alguma vez do agente *BitHead*? E quando o agente joga contra ele mesmo, qual o resultado?

3. Abra o código fonte do agente *BitHead*. Analise os métodos desenvolvidos, pois estes podem ser uma boa referência para a implementação da AAI. Utilize os documentos da pasta *docs* dentro de *PASTA_LAB3* como referência para as funções. Depois de analisado o código, responda:

   a) Porque a variável *bestValue* é inicializada com valor *-10* no caso das chamadas do método *maxValue* e com valor *+10* nas chamadas do método *minValue*?

   b) Explique a chamada a seguir, que aparece no corpo do método *minValue*. Para que se utiliza o método *copy* do objeto *game*, passando o número do oponente como parâmetro? Uma chamada semelhante é feita no corpo do método *maxValue*, explique a diferença entre estas duas chamadas.

   ```
   int value = maxValue(game.copy(opponent(game)));
   ```

   c) Por que o método *minimax* retorna uma jogada (classe *Move*) enquanto os métodos *minValue* e *maxValue* retornam somente um valor inteiro? Estes métodos também não deveriam retornar uma jogada? Para explicar melhor sua resposta, você pode desenhar um estado qualquer do jogo da velha e mostrar em quais estados cada um destes algoritmos são empregados.

4. Jogar contra o agente *BitHead* da forma como este foi desenvolvido não é estimulante. Como você faria para modificar o grau de dificuldade oferecido pelo agente? Aponte no código fonte do agente *BitHead* onde você poderia aplicar a sua técnica. Para testá-la, crie um *package* com nome *ia.tictactoe.agents* e crie uma classe com nome *NotSoGoodAgent.java* dentro deste *package*. Copie para esta classe o código fonte do agente *BitHead*. Execute a mudança e teste a classe criada, fazendo-a jogar contra o *BitHead*. (Dica: Uma técnica muito simples pode ser empregada usando números aleatórios!).

5. Adicione ao código uma forma simplificada de *alpha-beta pruning*, considerando que os melhores resultados sempre serão +1 (para o max) e -1 (para o min). DICA: Você precisa colocar apenas um ***if*** (*expressão*) que executa um ***return*** caso *expressão* seja verdadeira.

BOM TRABALHO!