

UNISALESIANO
Centro Universitário Católico Salesiano *Auxilium*
Curso de Tecnologia em Sistemas para Internet

Célia Hirata Aoki

JOGO DA MEMÓRIA
PARA DISPOSITIVOS MÓVEIS

LINS – SP
2009

CÉLIA HIRATA AOKI

JOGO DA MEMÓRIA PARA DISPOSITIVOS MÓVEIS

Trabalho de Conclusão de Curso
apresentado à Banca Examinadora do
Centro Universitário Católico Salesiano
Auxilium, curso de Tecnologia em
Sistemas para Internet, sob a orientação
do Prof. M.Sc Anderson Pazin.

LINS – SP

2009

Aoki, Célia Hirata
A647j Jogo da memória para dispositivos móveis / Célia Hirata Aoki. --
Lins, 2009.
88p. il. 31cm.

Monografia apresentada ao Centro Universitário Católico Salesiano *Auxilium* – UNISALESIANO, Lins-SP, para graduação em Tecnologia em Sistemas para Internet, 2009
Orientador: Anderson Pazin

1. Dispositivos móveis. 2. Jogos. 3. J2ME. I Título.

CDU 004

CÉLIA HIRATA AOKI

JOGO DA MEMÓRIA PARA DISPOSITIVOS MÓVEIS

Monografia apresentada ao Centro Universitário Católico Salesiano *Auxilium*, para obtenção do título de Tecnólogo em Sistemas para Internet.

Aprovada em: **16 / 06 /2009**

Banca Examinadora:

Prof(a) Orientador(a): **Prof. M.Sc. Anderson Pazin**

Titulação: **Mestre em Ciência da Computação pela Universidade Federal de São Carlos (UFSCar).**

Assinatura: _____

1º Prof(a): **Prof. M.Sc. Luiz Eduardo Cortes Bergamo**

Titulação: **Mestre em Ciência da Computação pela Universidade Federal de São Carlos (UFSCar).**

Assinatura: _____

2º Prof(a): **Prof. M.Sc. Ricardo Yoshio Horita**

Titulação: **Mestre em Ciência da Computação pela Universidade Federal de São Carlos (UFSCar)**

Assinatura: _____

DEDICATÓRIA

Ao querido Jeferson Fernando Martins (Son),
pela alegria e incentivo.
Saudades sempre...

AGRADECIMENTOS

A Deus,
pelo dom da vida e capacidade de perseverar.

Aos meus pais, irmãos e namorado,
pelo amor, apoio e compreensão.

Ao professor e orientador Prof. M.Sc. Anderson Pazin,
pela competência, disponibilidade e paciência.

Ao Coordenador Luiz Eduardo Cortes Bergamo e a todos os professores do
Curso,
pela dedicação, conhecimento e comprometimento.

Ao Unisalesiano, na pessoa do Reitor Pe. Paulo Fernando Vendrame,
pelo auxílio, incentivo e valorização.

Aos colegas de classe, em especial aos amigos Vagner e Vinícius,
pela alegria, amizade, tolerância e colaboração.

A todos, que de alguma forma, contribuíram para realização deste trabalho.

RESUMO

As fortes expansões do mercado de desenvolvimento de *softwares*, atrelados aos avanços tecnológicos dos dispositivos móveis, tornam a plataforma *Java Micro Edition* (J2ME) ideal para o desenvolvimento de aplicações para dispositivos com baixo poder de processamento. A tecnologia J2ME vem se destacando por oferecer um conjunto de especificações e recursos destinados a solucionar os problemas de desenvolvimento de aplicações para esses tipos de dispositivos. Assim, no presente trabalho, foi desenvolvido um jogo da memória para dispositivos móveis, utilizando a tecnologia J2ME, com o intuito de estudar de forma prática e mais detalhada esta plataforma. O uso desta aplicação pode ser, para iniciantes, um recurso facilitador no aprendizado e entendimento dos conceitos e funcionalidades da tecnologia J2ME.

Palavras-chave: Dispositivos móveis. Jogos. J2ME.

ABSTRACT

The strong expansion of the market for software development, coupled to technological advances of mobile devices makes Micro Edition Java platform (J2ME) ideal for the developing applications for devices with low processing power. The J2ME technology has been increasing because by to offer a set of specifications and resources destined to resolve the problems of the development of applications to these types of devices. Thus, in this work, was developed a game of memory to mobile devices using the J2ME technology in order to study in a practical and detailer is this platform. The use of this application may be, to starters, a resource facilitator in learning and understanding of the concepts and features of the J2ME technology.

Keywords: Mobile devices. Games. J2ME.

LISTA DE FIGURAS

Figura 1. Diagrama de Casos de Uso	16
Figura 2. Diagrama de Sequência.....	17
Figura 3. Diagrama de Colaboração	18
Figura 4. Diagrama de Classes	19
Figura 5. Diagrama de Componentes	19
Figura 6. Diagrama de Atividades	20
Figura 7. Diagrama de Temporização	21
Figura 8. Arquitetura do perfil MIDP	26
Figura 9. Jogo Minero's Snake	29
Figura 10. Jogo Auto-Estrada.....	30
Figura 11. Jogo Paranoid	30
Figura 12. Jogo da memória para crianças.....	31
Figura 13. Jogo da memória com relógio.....	32
Figura 14. Jogo da memória de letras e cores.....	32
Figura 15. Jogo da memória para iniciantes	33
Figura 16. Jogo da memória dos países	33
Figura 17. Diagrama de Use Case 01 – Iniciar Partida.....	37
Figura 18. Diagrama de Use Case 02 – Jogar Partida	38
Figura 19. Diagrama de Use Case 03 – Escolher Nível de Dificuldade	38
Figura 20. Diagrama de Use Case 04 – Visualizar Ranking.....	39
Figura 21. Diagrama de Use Case 05 – Pausar Partida.....	39
Figura 22. Diagrama de Use Case 06 – Reiniciar Partida	40
Figura 23. Diagrama de Use Case 07 – Encerrar Partida	40
Figura 24. Diagrama de Use Case 08 – Sair do Jogo	41
Figura 25. Diagrama de Use Case – Resumo do Jogo	41
Figura 26. Diagrama de Classes	42
Figura 27. Diagrama de Sequência – Iniciar Partida	43
Figura 28. Diagrama de Sequência – Jogar Partida	44
Figura 29. Diagrama de Colaboração – Iniciar Partida	45
Figura 30. Diagrama de Colaboração – Jogar Partida.....	46
Figura 31. Diagrama de Componentes	47

Figura 32.	Interface de “abertura”.....	48
Figura 33.	Interfaces do início do jogo	49
Figura 34.	Interface do jogo.....	50
Figura 35.	Interface da finalização da partida	51
Figura 36.	Página disponível para o <i>download</i> da IDE Eclipse	59
Figura 37.	Execução da IDE Eclipse.....	60
Figura 38.	Ambiente Eclipse.....	60
Figura 39.	Página disponível para o <i>download</i> do <i>Sun Wireless Toolkit</i>	61
Figura 40.	Início da instalação do <i>Sun Wireless Toolkit</i>	61
Figura 41.	Início da instalação do <i>Sun Wireless Toolkit</i>	62
Figura 42.	Tela para configurar a localização da JVM.....	62
Figura 43.	Tela para configurar a localização da instalação do WTK.....	63
Figura 44.	Início da instalação do <i>plugins</i> Eclipse ME	63
Figura 45.	Tela para escolher o tipo de instalação	64
Figura 46.	Tela para definir o nome do repositório e da url	64
Figura 47.	Tela para finalização e instalação dos <i>plugins</i>	65
Figura 48.	Página disponível para fazer o <i>download</i> dos <i>plugins</i>	65
Figura 49.	Página disponível para fazer o <i>download</i> dos <i>plugins</i>	66
Figura 50.	Acesso a tela de configurações	66
Figura 51.	Tela de configurações	67
Figura 52.	Tela para criação do novo projeto.....	67
Figura 53.	Tela para criação da <i>MIDlet</i>	68
Figura 54.	Configuração da <i>MIDlet</i>	68
Figura 55.	Nova <i>MIDlet</i> criada – com os principais <i>imports</i> , seus métodos ... básicos e o método construtor.....	69
Figura 56.	Código da <i>MIDlet</i>	70
Figura 57.	Processo para execução da <i>MIDlet</i>	70
Figura 58.	Resultado da execução da <i>Midlet</i>	71
Figura 59.	Tela de criação do <i>Midlet Suite</i>	75
Figura 60.	Hierarquia da classe <i>Displayable</i>	76
Figura 61.	Tela com as cartas desenhadas	85
Figura 62.	Tela do programa Motomidman	87

LISTA DE QUADROS

Quadro 1. Lista de Casos de Uso	37
Quadro 2. Pacotes da configuração CLDC	72

LISTA DE SIGLAS E ABREVIATURAS

API:	<i>Application Programming Interface</i> (Interface de Programação de Aplicativos)
CDC:	<i>Connected Device Configuration</i> (Configuração de Dispositivos Conectados)
CLDC:	<i>Connected Limited Device Configuration</i> (Configuração de Dispositivos Conectados Limitados)
GPS:	<i>Global Positioning System</i> (Sistema de Posicionamento Global)
IDE:	<i>Integrated Development Environment</i> (Ambiente de Desenvolvimento Integrado)
JVM:	<i>Java Virtual Machine</i> (Máquina Virtual Java)
J2EE:	<i>Java 2 Enterprise Edition</i>
J2ME:	<i>Java 2 Micro Edition</i>
J2SE:	<i>Java Standard Edition</i>
KVM:	<i>Kilobyte Virtual Machine</i>
MHz:	<i>Megahertz</i>
MIDP:	<i>Mobile Information Device Profile</i> (Perfil de Dispositivo de Informação Móvel)
MV:	<i>Virtual Machine</i> (Máquina Virtual)
OOSE:	<i>Object-Oriented Software Engineering</i>
OMT:	<i>Object Modeling Technique</i>
PDA:	<i>Personal Digital Assistant</i> (Assistente Digital Pessoal)
RGB:	<i>Red, Green, Blue</i> (Vermelho, Verde, Azul)
SMS:	<i>Short Message Service</i> (Serviço de Mensagem Curta)
UML:	<i>Unified Modeling Language</i> (Linguagem de Modelagem Unificada)
WAP:	<i>Wireless Application Protocol</i> (Protocolo para Aplicações sem Fio)

SUMÁRIO

INTRODUÇÃO	12
1 FUNDAMENTOS CONCEITUAIS	14
1.1 UML – <i>UNIFIED MODELING LANGUAGE</i>	14
1.2 PLATAFORMAS TECNOLÓGICAS	21
1.2.1 JAVA	21
1.2.2 J2ME - <i>JAVA MICRO EDITION</i>	23
2 LEVANTAMENTO DE REQUISITOS	27
2.1 ANÁLISE DE MERCADO	27
2.1.1 JOGOS PARA COMPUTADOR	27
2.1.2 JOGOS PARA DISPOSITIVOS MÓVEIS	27
2.2 TIPOS DE JOGOS PARA DISPOSITIVOS MÓVEIS	29
2.2.1 MINERO'S SNAKE	29
2.2.2 AUTO-ESTRADA	29
2.2.3 PARANOID	30
2.3 JOGO DA MEMÓRIA	30
2.4 MODELOS DE JOGOS DA MEMÓRIA	31
2.5 DOCUMENTOS DE REQUISITOS	34
2.5.1 VISÃO GERAL DO SISTEMA	34
2.5.2 REQUISITOS FUNCIONAIS	34
2.5.3 REQUISITOS NÃO FUNCIONAIS	36
3 ANÁLISE ORIENTADA A OBJETOS	37
3.1 LISTA DE CASOS DE USO	37
3.2 DIAGRAMA DE CASOS DE USO	37
3.2.1 INICIAR PARTIDA	37
3.2.2 JOGAR PARTIDA	38

3.2.3	ESCOLHER NÍVEL DE DIFICULDADE	38
3.2.4	VISUALIZAR RANKING	39
3.2.5	PAUSAR PARTIDA	39
3.2.6	REINICIAR PARTIDA	40
3.2.7	ENCERRAR PARTIDA	40
3.2.8	SAIR DO JOGO	41
3.2.9	RESUMO DO JOGO	41
3.3	DIAGRAMA DE CLASSES	42
4	PROJETO ORIENTADO A OBJETOS	43
4.1	DIAGRAMAS DE SEQUÊNCIA	43
4.2	DIAGRAMA DE COLABORAÇÃO	45
5	IMPLEMENTAÇÃO ORIENTADA A OBJETOS	47
5.1	DIAGRAMA DE COMPONENTES	47
5.2	LAYOUT DE TELAS	48
	CONCLUSÃO	53
	REFERÊNCIAS	55
	ANEXOS	58

INTRODUÇÃO

A indústria de celulares está passando por mudanças significativas. Os aparelhos que, até pouco tempo atrás, só realizavam chamadas estão se tornando verdadeiros computadores de bolso, pois combinam a capacidade de navegação pela Internet com uma série extensa de serviços, como máquina fotográfica, filmadora, posicionamento GPS etc. Na prática, a indústria de celulares está mudando seu foco de hardware para software.

O mercado de desenvolvimento de aplicações para dispositivos móveis tende a crescer 102% ao ano no mundo, nos próximos cinco anos. (SPOSITO, 2007). Uma das possibilidades para esse segmento é o desenvolvimento de jogos que, segundo Barboza; Silva (2007), é um mercado amplo e com forte tendência de crescimento, apresentando-se como uma alternativa para participação dos pequenos desenvolvedores, uma vez que os custos de produção de jogos para computadores pessoais e videogame são elevados.

Ainda de acordo com Barboza; Silva (2007), outro nicho que vem se expandindo de forma promissora, e também possibilitando a entrada dos pequenos desenvolvedores no mercado, é a publicidade nos jogos de dispositivos móveis. Este tipo de publicidade é uma estratégia bastante satisfatória e pode trazer vantagens como:

- a) custear a produção do jogo, através de parcerias;
- b) atingir um público variado e amplo;
- c) ser aplicada em diversos segmentos.

Neste contexto de desenvolvimento de software, a linguagem Java oferece uma plataforma de desenvolvimento chamada *Java Micro Edition* (J2ME) que é totalmente direcionada a pequenos dispositivos tais como: celular, *pager*, *palm*, entre outros. Essa plataforma pode ser definida como um conjunto de tecnologias e especificações, com características para atender as necessidades que o desenvolvimento de software para dispositivos móveis impõe: baixo poder de processamento e pouca memória disponível. (FONSECA, 2005)

Sendo o J2ME uma tecnologia atual, que oferece inúmeras possibilidades de utilização, o presente trabalho tem como objetivo apresentar

a funcionalidade e a aplicabilidade dessa tecnologia. Para isso, será desenvolvido um Jogo da Memória para dispositivos móveis.

Optou-se por desenvolver uma aplicação prática, que faz uso da tecnologia J2ME, pelo fato de considerar que este pode ser um meio bastante eficiente de aprendizagem.

O Jogo da Memória é um tipo clássico de jogo, formado por cartas que apresentam uma imagem em um dos lados, e estas se repetem em duas cartas diferentes, formando um par. O objetivo principal do jogo é que todos os pares sejam encontrados com a menor quantidade de jogadas possíveis. Assim, em uma simples brincadeira, a capacidade de memorização das pessoas é estimulada e de maneira bastante agradável.

O desafio de desenvolver um trabalho, onde o aprendizado ultrapassa os limites da sala de aula, foi um ponto relevante para escolha deste projeto, bem como oportunidade de estudar a tecnologia J2ME de forma mais aprofundada. Segundo Taurion (2008), 85% dos aparelhos celulares utilizam tal tecnologia, e esta tem sido vista como uma tendência de mercado.

1 FUNDAMENTOS CONCEITUAIS

1.1 UML – *Unified Modeling Language*

A documentação de software permite detalhar por meio de modelos, diagramas e textos todo o funcionamento de um sistema. Embora essa atividade seja de grande importância no processo de desenvolvimento, ela é pouco valorizada e executada. (BICALHO, 2008)

Um dos problemas mais frequentes, gerados pela falta de documentação, é a falha na comunicação entre as partes interessadas, o que acaba por ocasionar muitos erros e grande perda de tempo. Uma das características da documentação é diminuir o retrabalho.

“A modelagem é um meio de documentar idéias, relacionamentos, decisões e requisitos numa notação bem definida que pode ser aplicada a muitos lugares diferentes.” (FREITAS, 2008).

O surgimento da linguagem de programação orientada a objeto trouxe como consequência a necessidade de um método de modelagem adequado a ela. Assim, entre as décadas de 70 e 80, aproximadamente, surgem linguagens de modelagem orientadas a objeto, com métodos alternativos de análise e projeto. Entre estas novas linguagens de modelagem destacam-se: Booch, OOSE (*Object-Oriented Software Engineering*) de Jacobson, e o OMT (*Object Modeling Technique*) de Rumbaugh. Estas se tratavam de linguagens completas, porém, com limitações que impediam de atender inteiramente às necessidades de cada projeto.

- a) O método Booch destacava-se durante as fases de projeto e construção de sistemas;
- b) O OOSE fornecia excelente suporte para captura de requisitos, a análise e o projeto em alto nível;
- c) O OMT-2 era mais útil com a análise e sistemas de informações com uso de dados. (NOGUEIRA, 2005).

A UML surge da união destes métodos, onde o melhor que cada um oferecia foi reunido, visando desta forma fornecer ao mercado uma linguagem padrão, que fosse de fácil entendimento a todos. O fato de deixar aberta aos desenvolvedores possibilitou que estes pudessem criar seu próprio método de trabalho.

A UML foi projetada para auxiliar aqueles que participam da atividade de desenvolvimento de *software* a construir modelos que permitam visualizar o sistema, especificar a estrutura e o comportamento deste, construí-lo e documentar as decisões tomadas durante o processo. (SCOTT, 2003, p. 19).

Basicamente a UML é utilizada para modelar as fases de um sistema, desde os primeiros contatos até a geração do código. Contudo, também pode ser aplicada em sistemas mecânicos, de engenharia em geral e auxiliar na organização de processos de uma organização. (NOGUEIRA, 2005).

Dentre as várias aplicações da linguagem UML podem ser consideradas como as mais comuns: o projeto de software, a comunicação de processo de software ou de negócios, a documentação detalhada sobre um sistema para requisitos ou análise, a documentação de um sistema ou processo. (FREITAS, 2008).

Ela pode ser definida como um modelo de linguagem e não como um método, pois um método é composto de um modelo de linguagem e um processo. A UML define uma notação e um meta-modelo, onde a notação representa a sintaxe da linguagem composta por elementos gráficos e um meta-modelo é um diagrama de classe.

Segundo Esmin (2008), as partes que compõem a UML são:

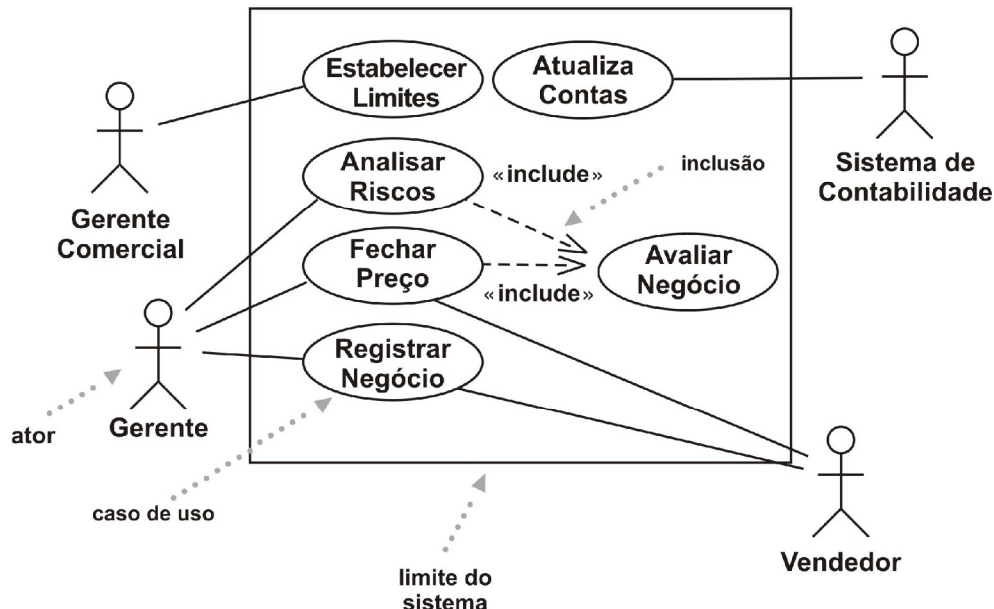
- a) as visões, que mostram os diferentes aspectos do sistema que está sendo modelado através de diagramas;
- b) os mecanismos gerais, comentários suplementares, informações ou semântica sobre os elementos que compõem os modelos;
- c) os diagramas, gráficos que descrevem o conteúdo em uma visão, que podem ser utilizados em combinação. São nove tipos de diagramas: Diagrama de Estados, Diagrama de Objetos (Colaboração), Diagrama de Processo (Desenvolvimento), Diagrama de Módulos (Componentes), Use Case, Subsistemas (Package), Diagrama de Interações, Mini-Especificação, Diagrama de Estados.

Bons diagramas podem auxiliar na transmissão de idéias de um projeto e também facilitar sua compreensão, entretanto deve-se conhecer e entender o que cada diagrama pode oferecer e saber como aplicá-lo da melhor forma, e assim obter um resultado eficiente, pois pode ocorrer de um conceito ter a

possibilidade ser expresso por mais de um tipo de diagrama. (FREITAS, 2008).

Entre os vários tipos de diagramas podemos destacar alguns, como o Diagrama *Use Case* (Caso de Uso), utilizado para identificar como o sistema se comporta nas várias situações que podem ocorrer durante sua operação. Os componentes deste diagrama são os atores e os *Use Case*; o ator possui comportamento como uma pessoa, um sistema ou uma organização, por exemplo, um vendedor. Os *use cases* são os cenários relacionados, que descrevem um ator utilizando o sistema para atingir seu objetivo. (LARMAN, 2007).

A Figura 1 apresenta um exemplo de Diagrama de *Use Case*, que trata da compra de um produto. Compõem este diagrama: os atores, (o gerente comercial, o vendedor, gerente e o sistema de contabilidade), os casos de uso (registrar negócio, fechar preço, analisar riscos, estabelecer limites, atualiza contas e avaliar negócio) e relacionamento entre eles.



Fonte: FOWLER, 2005, p. 107

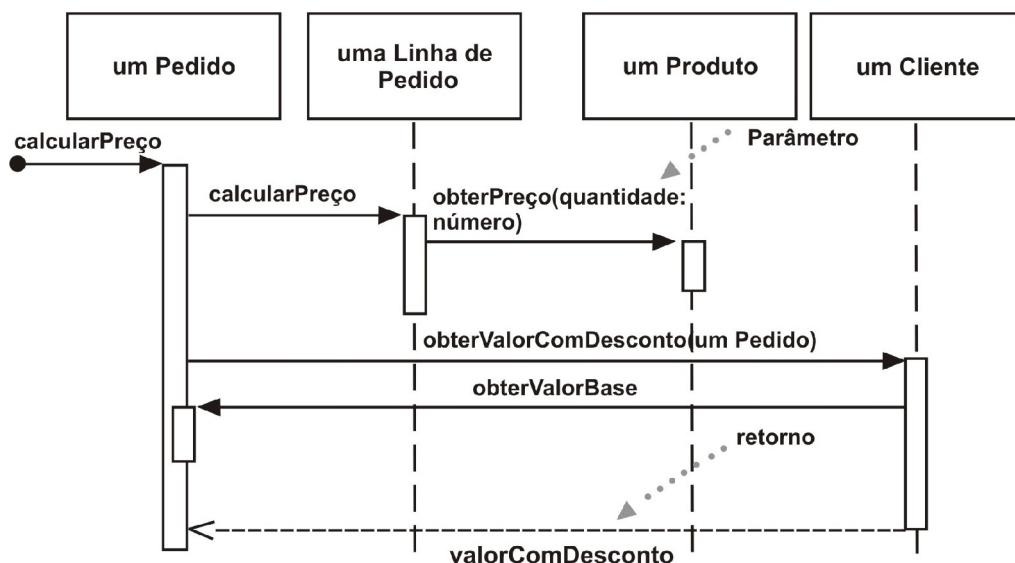
Figura 1. Diagrama de Casos de Uso

Os diagramas de interação são grupos de objetos que auxiliam em algum comportamento. Dentre os vários tipos de diagramas de interação

definidos pela UML o mais comum é o diagrama de sequência, que pode ser definido como aquele que captura o comportamento de um único cenário, apresenta a interação entre os objetos ao longo do tempo. (FOWLER, 2005).

Um exemplo do diagrama de sequência pode ser visto na Figura 2, que descreve a interação entre objetos para o cálculo do preço de um produto; neste modelo é utilizado o controle distribuído, onde o processamento é dividido em vários participantes.

Um pedido solicita o cálculo do seu próprio preço para uma linha de pedido. Esta transmite o cálculo para o produto para obter o preço deste. O pedido chama um método do cliente para obter o preço com desconto, porém, para isto o cliente necessita de obter o valor base do pedido e desta forma poder retornar o valor com desconto.

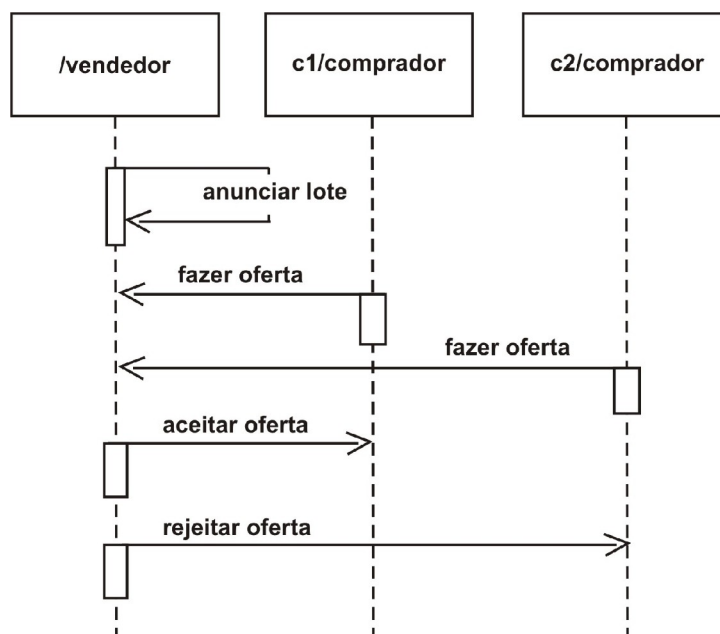


Fonte: FOWLER, 2005, p. 69

Figura 2. Diagrama de Sequência

Diagramas de colaboração mostram como os objetos interagem ao realizar o comportamento de um caso de uso. São considerados uma importante fonte de informação para determinar interfaces e responsabilidades de cada classe, são adequados às atividades de análise e representam melhores interações simples com pouco número de objetos. É composto de objetos e instâncias de ator, além de links e mensagens e descreve o que acontece com os objetos participantes.

Na Figura 3, há o exemplo de um Diagrama de colaboração, que descreve a interação dos objetos vendedor e comprador no processo de um leilão.



Fonte: FOWLER, 2005, p. 137

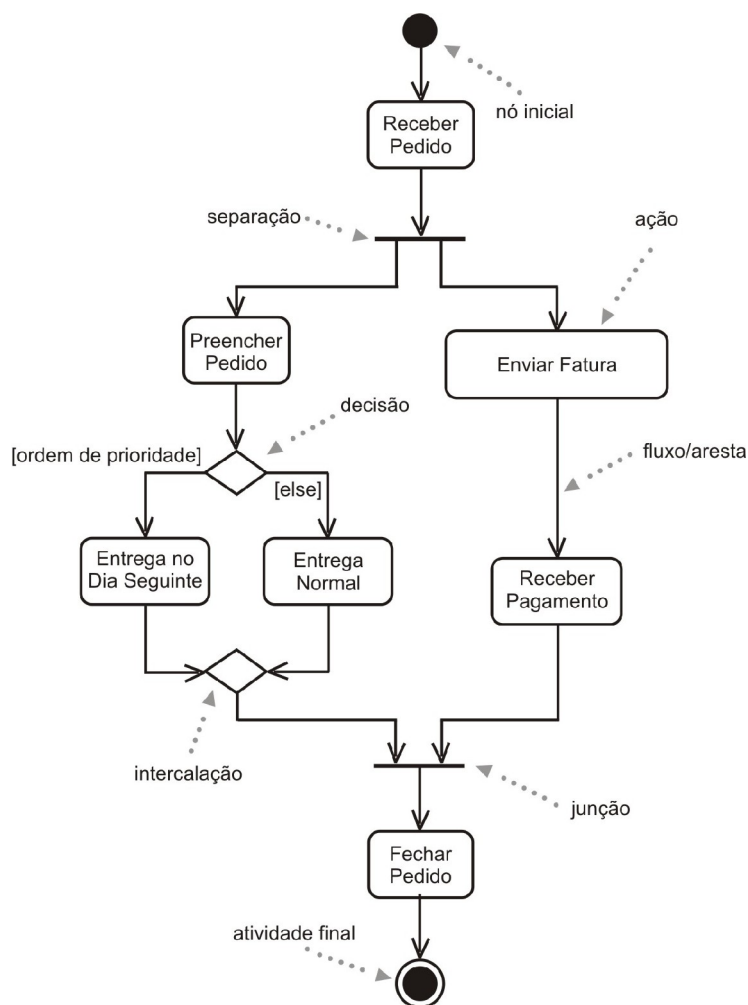
Figura 3. Diagrama de Colaboração

O Diagrama de Classes é o mais utilizado, pois demonstra a estrutura estática das classes de um sistema onde estas representam objetos que são gerenciados pela aplicação modelada. Uma classe pode se relacionar com outra de diversas maneiras: associação (conectadas entre si), dependência (uma classe depende ou usa outra classe), especialização (uma classe é uma especialização de outra classe), ou em pacotes (classes agrupadas por características similares). Todos estes relacionamentos são mostrados no diagrama de classes juntamente com as suas estruturas internas, que são os atributos e operações. (FOWLER, 2005).

A Figura 4 mostra um modelo de Diagrama de Classes, que trata do processamento de pedidos. É composto pelas classes, que são divididas em três partes: nome, atributos e operações, e também pelos relacionamentos existentes entre elas (associação e generalização) e as devidas cardinalidades.

Os Diagramas de Atividades podem ser definidos como uma técnica para descrever lógica de procedimentos, processo de negócio e fluxo de trabalho. Assemelham-se ao papel do fluxograma com a diferença de que os diagramas suportam comportamento paralelo. Também permite que quem está seguindo o processo determine a ordem a ser seguida. (FOWLER, 2005)

A Figura 6 apresenta um exemplo de Diagrama de Atividades, que trata do processo de um pedido. Preencher pedido, enviar fatura e ações subsequentes são comportamentos paralelos.

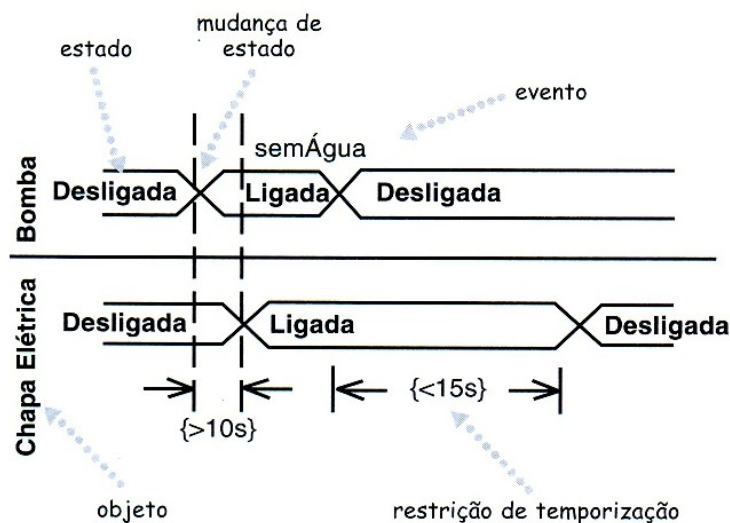


Fonte: FOWLER, 2005, p. 119

Figura 6. Diagrama de Atividades

Diagramas de Temporização são outra forma de diagrama de interação, onde o foco está nas restrições de temporização, para um único objeto ou para vários. (FOWLER, 2005).

A Figura 7 mostra maneiras alternativas de apresentar restrições de temporização.



Fonte: FOWLER, 2005, p. 141

Figura 7. Diagrama de Temporização

1.2 Plataformas Tecnológicas

1.2.1 Java

As linguagens de programação podem ser consideradas instrumentos fundamentais para o desenvolvimento de aplicações. Desta forma, o profissional não deve restringir seu conhecimento a somente uma linguagem, e sim se aprofundar, buscando descobrir as vantagens e desvantagens oferecidas por elas, tendo em vista que não há melhor ou pior, e sim a que mais irá se adequar ao seu projeto (MATTOS, 2004).

Assim, a escolha de uma linguagem, para o desenvolvimento de uma determinada aplicação, não deve ser norteadas somente pelo conhecimento dela, mas também pelo conhecimento da aplicação e suas necessidades (LEITE, 2006). Outro fator que tem grande influência na escolha de uma linguagem de programação é o ambiente em que ela poderá ser utilizada.

A determinação da linguagem a ser empregada é responsabilidade do programador, que deve ter consciência de que a melhor linguagem não é

aquela que produz a melhor tela ou que está na “moda”, e sim aquela que aperfeiçoa a aplicação, satisfazendo plenamente o cliente e sobre a qual tem pleno domínio (LEITE, 2006).

Java é uma linguagem de programação orientada a objeto, de alto nível, com sintaxe similar a do C++, que foi lançada em 1991, pela Sun Microsystem. Surgiu a partir de um projeto, chamado “Projeto Green”, que foi desenvolvido por um grupo de pesquisadores, onde o objetivo não era simplesmente criar uma linguagem de programação convencional, mas sim determinar uma nova tendência do mundo digital.

A princípio foi destinada à programação de dispositivos eletrônicos inteligentes voltados ao consumo popular, pois a Sun acreditava na computação embarcada. Atualmente esta é uma realidade viável, mas naquele momento não houve interesse e a linguagem Java não obteve bons resultados em seus propósitos iniciais (JAVA, 2008). Seu advento veio somente com o surgimento da internet (SILVEIRA, 2003).

A linguagem Java foi rapidamente adotada e, em 2004, atingiu a marca de três milhões de desenvolvedores em todo mundo e continua crescendo. Hoje é considerada uma referência no mercado de desenvolvimento de *softwares* e tem seu ambiente de execução presente em diversas aplicações como: *web*, *desktop*, servidores, mainframes, jogos, aplicações móveis, chips de identificação entre tantos outros. (JAVA, 2008)

Diferente das outras linguagens convencionais, que são compiladas para código nativo da máquina, a linguagem Java é compilada para um *bytecode* (*bytecode* ou código em bytes mais conhecido pelos caracteres 010010) que é executado por uma máquina virtual (MV).

A máquina virtual pode ser definida como aquela que, implementada através de um software, executa programas como um computador real. Não é composta por um hardware, é um computador fictício (JAVA, 2008)

A JVM e os *bytecodes* são peças de grande relevância dentro da linguagem Java, principalmente no que se refere à portabilidade. A JVM é o mecanismo que permite que o código Java seja executado em qualquer plataforma, isto é possível devido aos *bytecodes*, uma espécie de codificação, que traduz o que foi escrito no programa para um formato que a JVM entenda e seja capaz de executar, pois os *bytecodes* são constituídos da mesma forma,

independente do Sistema Operacional. (ALECRIM, 2005)

De acordo com Pamplona (2008), Java pode ser dividida em três grandes edições:

- a) *Java Standard Edition* (J2SE), que é a tecnologia voltada para computadores com poder de processamento e memória considerável;
- b) *Java 2 Micro Edition* (J2ME), tecnologia voltada para dispositivos móveis, com limitação de processamento ou memória;
- c) *Java 2 Enterprise Edition* (J2EE), tecnologia para aplicações corporativas, que podem estar na internet ou não.

A API da linguagem vem aumentando gradualmente, disponibilizando várias bibliotecas, e a cada versão apresenta modificações importantes. (SILVEIRA, 2003).

Outras características da linguagem Java que podem ser citadas como principais:

- a) Portabilidade, independência de plataforma;
- b) Recursos de rede, possui extensa biblioteca de rotinas;
- c) Segurança, pode executar programas via rede com restrições de execução.

Além de possuir facilidades para criação de programas distribuídos e multitarefa, desalocação de memória automática e carga dinâmica de código.

1.2.2 J2ME - *Java Micro Edition*

A plataforma J2ME, *Java Micro Edition*, foi projetada para possibilitar o desenvolvimento de aplicações destinadas aos dispositivos com baixo poder de processamento e com uma quantidade limitada de recursos, como celulares e *pagers*. As aplicações para estes dispositivos, antes da linguagem J2ME, eram escritas na linguagem nativa do dispositivo móvel, tornando-se um problema para programadores, uma vez que os mesmos teriam que aprender a linguagem específica de cada dispositivo.

Com a implementação do pacote de desenvolvimento J2ME, surge então um modo de programar seguro, portátil e fácil de dominar, isto sem mencionar que a comunidade de desenvolvedores da linguagem Java vem

crecendo a cada ano. (MUCHOW, 2006).

Os diversos tipos de equipamentos eletrônicos, disponíveis em nosso dia a dia, são capazes de executar a tecnologia J2ME, cada qual com sua capacidade computacional, que varia de aparelho para aparelho.

Aparelhos de maior porte são mais poderosos, uma vez que podem possuir processadores maiores, maior quantidade de memória, de disco e hardware. Já os dispositivos menores são mais restritos, têm acesso a uma fonte de energia limitada.

Essas diferenças de recursos computacionais podem gerar uma sub ou super utilização dos mesmos durante a execução de um aplicativo. Assim para que a linguagem J2ME pudesse suportar uma grande variedade de dispositivos, a SUN introduziu o conceito de configuração, que é um conjunto de bibliotecas, classes e APIs, que fornecem a funcionalidade básica para os dispositivos com características similares.

Essas configurações de modo geral são baseadas nos recursos que cada dispositivo apresenta como memória e poder de processamento. Atualmente são classificados em dois modos: “Configuração de Dispositivos Conectados”, ou simplesmente “CDC”, ou “Configuração de Dispositivos Conectados Limitados”, “CLDC”.

A Configuração de Dispositivos Conectados (CDC) apresenta as seguintes características: (MUCHOW, 2006)

- a) 512 Kilobyte (no mínimo) de memória para executar o Java;
- b) 256 Kilobyte (no mínimo) de memória para alocação de memória em tempo de execução;
- c) conectividade de rede, largura de banda possivelmente persistente e alta.

A Configuração de Dispositivos Conectados Limitados (CLDC) apresenta estas características:

- a) 128 Kilobyte de memória para executar o Java;
- b) 32 Kilobyte para alocação de memória em tempo de execução;
- c) interface restrita com o usuário;
- d) baixo poder, normalmente alimentado por bateria;
- e) conectividade de rede normalmente dispositiva sem fio com largura de banda baixa e acesso intermitente. (MUCHOW, 2006, p.4).

Todo dispositivo tem uma configuração própria que é única, por exemplo, telefones celulares ou PDAs se encaixarão na configuração de um CLDC.

Ainda entre os aparelhos com recursos limitados, ocorre outra sub-classificação, pois entre eles também existem diferenças, assim é introduzido o conceito para tratar esta variação de recursos, o Perfil.

O perfil trata-se de uma extensão da camada de Configuração que fornece somente as bibliotecas necessárias para um dispositivo em particular, assim cada perfil está associado a somente uma configuração. (JOHNSON, 2008).

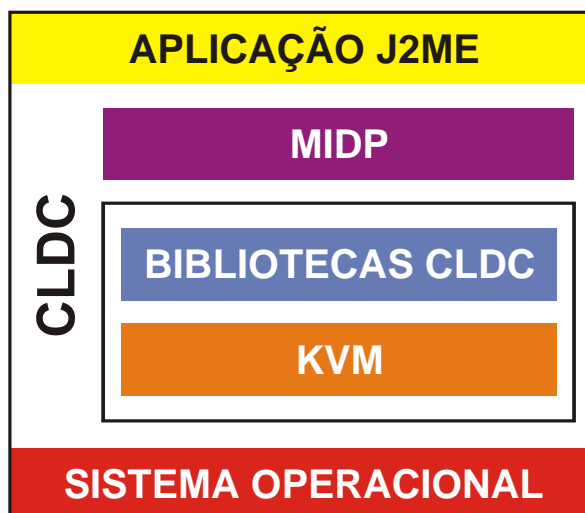
O perfil da configuração CLDC que dá suporte à maioria dos dispositivos móveis atuais é o MIDP (*Mobile Information Device Profile* – Perfil de Configuração de Dispositivos Móveis). Segundo Johnson (2008), o MIDP apresenta diversas funcionalidades como:

- a) suporte aos protocolos de rede do tipo HTTP e sockets;
- b) reprodução de multimídia;
- c) definição de formulários e itens;
- d) APIs para jogos;
- e) suporte ao sistema de cores RGB e validação de permissões de segurança;
- f) assinaturas virtuais.

Visando adaptar a máquina virtual às condições dos dispositivos móveis, a Sun disponibilizou um tipo de máquina virtual, a Kilobyte Virtual Machine ou simplesmente KVM.

Projetada para manipular a linguagem dentro dos dispositivos que possuem recursos limitados (CLDC). Sua exigência é de apenas de 40 a 80 *kilobyte* de memória e de 20 a 40 *kilobyte* de memória dinâmica, podendo ser executada em processadores de 16 Bit com frequências de apenas 25 MHz. (MUCHOW, 2006).

A Figura 8 trata da representação da arquitetura do perfil MIDP, que tem como base o Sistema Operacional, hospedeiro. Acima está a KVM (*Kilobyte Virtual Machine*), a máquina virtual que atende as especificações exigidas pela configuração CLDC. Em seguida, estão as bibliotecas CLDC, seguindo a hierarquia encontra-se o MIDP, perfil projetado para fornecer APIs mais específicas e, na última camada da arquitetura, está a aplicação.



Fonte: Johnson, 2008

Figura 8. Arquitetura do perfil MIDP

Para fornecer um ambiente completo, a cada categoria de dispositivos, a Sun desenvolveu o *Java Wireless Toolkit* (Java ME Downloads, 2009), também conhecido como *J2ME Wireless Toolkit* que nada mais é do que um conjunto de configuração e perfil.

O Java Wireless Toolkit fornece ao programador a MIDP que facilita a busca de quais bibliotecas são necessárias para desenvolver num determinado dispositivo. Também oferece um emulador¹ a bordo para que possa visualizar a aplicação e garantir que não haja erros.

Desta forma, os programadores não levam um longo tempo buscando bibliotecas para tais dispositivos e reduzem o espaço ocupado por J2ME ao mínimo.

Com todos esses recursos, a J2ME torna-se uma linguagem eficiente e prática em que o programador pode desenvolver seus aplicativos com facilidade, focando apenas em escrever o aplicativo com maior eficiência.

¹ “...é um software que reproduz as funções de um determinado ambiente, a fim de, permitir a execução de outros softwares sobre ele.” (WIKIPÉDIA, 2009).

2 LEVANTAMENTO DE REQUISITOS

2.1 Análise de Mercado

2.1.1 Jogos para Computador

Jogos são formas de entretenimento derivados de um conjunto de regras, tipicamente com um objetivo conhecido a ser alcançado. Podem ser na forma de atividades físicas, mentais, ou uma mistura dos dois. Também podem ser classificados como coletivos, individuais, ou competitivos.

Atualmente o conceito de jogos está correlacionado a jogos eletrônicos que fazem uso de computadores ou outros dispositivos eletrônicos.

Neste formato de jogo, a diversão é alcançada plenamente, pois a versatilidade quanto aos tipos de jogos que podem ser executados e aos tipos de públicos que podem atender são bastante amplos. Atualmente há jogos voltados exclusivamente para computadores, desenhados e programados de acordo com os recursos e ferramentas oferecidos por eles. Os jogos podem ser desde aqueles com recursos mais simples, como Paciência, Tetris, Campo Minado até os complexos com efeitos visuais 3D.

O uso da rede de computadores juntamente com o desenvolvimento de jogos mais complexos, impulsionou de forma bastante significativa a expansão dos jogos para computadores, uma vez que permitiu a participação de várias pessoas em um mesmo jogo, tornando-o mais competitivo e possibilitando uma interatividade que não exige que os jogadores estejam no mesmo local físico. (JOGO, 2008)

2.1.2 Jogos para Dispositivos Móveis

Segundo Johnson (2008), a computação móvel pode ser definida como aquela que possibilita aos usuários acesso a algum tipo de serviço independente de sua localização. Dentre os tipos de dispositivos que fazem uso da computação móvel pode-se citar: notebooks, PDAS, telefones celulares, entre outros.

Com a popularização dos dispositivos móveis, o mercado de desenvolvimento de jogos foi impulsionado de maneira significativa (BARBOZA;

SILVA, 2007). No Brasil já existem algumas empresas especializadas neste setor, que produzem também para outros países. (CORASSA, 2008)

Segundo Santos (2006), esses tipos de dispositivos apresentam-se cada vez mais evoluídos quanto à capacidade de processamento, armazenamento, integração e miniaturização de componentes eletrônicos, possibilitando assim o surgimento de um número maior e mais variado de recursos, que geram consequentemente o desenvolvimento de jogos de boa qualidade gráfica e com enredos envolventes.

Além do mercado crescente, o desenvolvimento de jogos voltados para dispositivos móveis oferece algumas vantagens sobre os jogos para computadores, como:

- a) poder ser produzido a partir de um baixo investimento;
- b) trabalhar com equipes menores;
- c) ser executado em menos tempo. (DESENVOLVIMENTO, 2006)

Entretanto, a elaboração de jogos para dispositivos móveis requer uma atenção especial em relação a limitações ainda não solucionadas. O fato de os dispositivos serem alimentados por baterias, e estas possuírem uma carga limitada, é recomendado que jogos muito longos sejam evitados, aqueles que exigem maior raciocínio do que habilidade dos dedos também são bem-vindos, uma vez que existe uma grande variedade de modelos de teclados e nem sempre são amigáveis. (FAGUNDES, 2008).

Em relação aos tipos de jogos para dispositivos móveis, estes podem classificados em:

- a) convencionais, que são jogados no próprio celular;
- b) de internet WAP, que pode ser jogado pela internet;
- c) de interação, via mensagens SMS.

Também podem ser divididos por categorias: esporte, ação, arcade, cartas, simuladores e *quiz*. Entre os mais populares destacam-se os de corrida (carro e moto), os jogos clássicos de cartas (truco, *blackjack*, pôquer) e futebol.

Os jogos para dispositivos móveis têm a facilidade de estar disponíveis em muitos sites, sendo muitos deles gratuitos. Normalmente o processo de instalação de um jogo em um dispositivo móvel se dá através de programas específicos de cada aparelho.

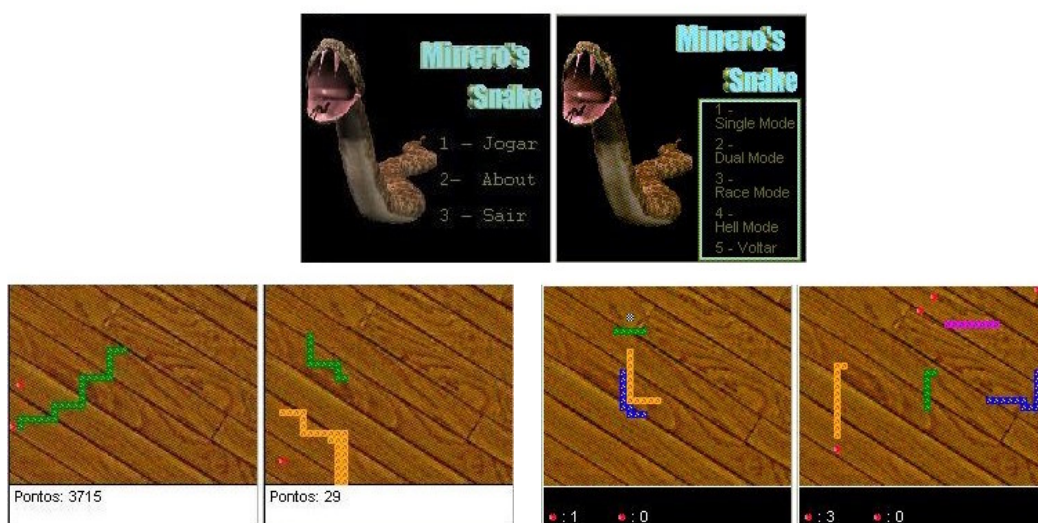
Contudo algumas considerações devem ser feitas ao escolher um jogo,

como a preocupação com a compatibilidade entre o tipo de jogo e o modelo do aparelho, pois alguns tipos de jogos requerem configurações específicas. Outra questão relevante é quanto à capacidade de armazenamento do dispositivo, algumas operadoras já disponibilizam em seus sites esta informação. (JOGOS, 2005).

2.2 Tipos de Jogos para Dispositivos Móveis

2.2.1 Minero's Snake

A Figura 9 ilustra o jogo Minero's Snake, que se trata de uma versão atualizada do famoso jogo da cobrinha, onde o jogador joga sozinho, ou no modo contra o computador.



Fonte: JOGOS, 2008

Figura 9. Jogo Minero's Snake

2.2.2 Auto-Estrada

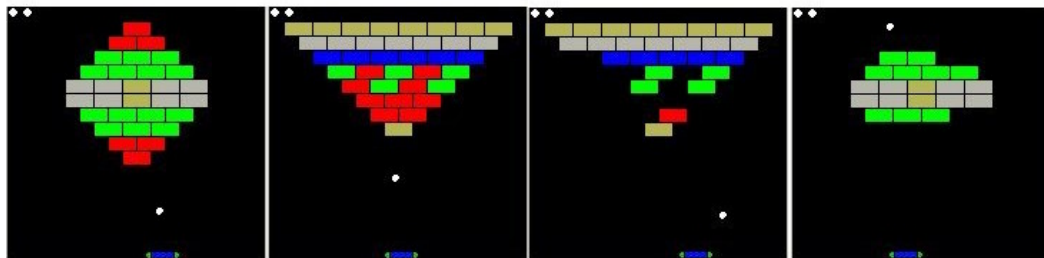
Na Figura 10, é exibido o jogo Auto-Estrada, onde o objetivo principal é chegar à primeira posição em uma Auto-Estrada ultracongestionada. Além do desafio de manobrar entre tantos carros, o jogador encontrará motoristas maldosos que tentarão atrapalhar a corrida.



Fonte: JOGOS, 2008

Figura 10. Jogo Auto-Estrada

2.2.3 Paranoid



Fonte: JOGOS, 2008

Figura 11. Jogo Paranoid

A Figura 11 apresenta um jogo clássico do PC, o Paranoid, cujo objetivo é rebater uma pequena bolinha para destruir todos os tijolinhos existentes.

2.3 Jogo da Memória

O jogo da memória é um clássico, formado por cartas que apresentam uma imagem em um dos lados, e cada imagem se repete em duas cartas diferentes, formando um par.

O jogo é iniciado com as cartas apresentadas com as figuras voltadas para baixo. A cada jogada o participante escolhe duas cartas, caso estas sejam iguais, as cartas permanecem voltadas para cima, caso contrário, as cartas

voltam à posição inicial, com as figuras voltadas para baixo. O objetivo do jogo é encontrar todos os pares no menor tempo possível ou no menor número de tentativas. Elevar a capacidade de memorização através do entretenimento é o objetivo subliminar do jogo.

2.4 Modelos de Jogos da Memória

Como o objetivo do jogo é bem definido, os modelos existentes atualmente se diferenciam pela forma de apresentação, com grau de dificuldade variado ou alguns recursos para facilitar a interação do jogador.

A dificuldade pode ser estabelecida pelo tamanho, tipo ou quantidade de imagens, pela forma de associação ou por algum outro fator escolhido.

Relógio para marcar o tempo da partida, um contador de jogadas, ranking de pontuação, definição do nome do jogador, botão para iniciar ou encerrar a partida são alguns dos recursos adicionais que podem aparecer em um jogo da memória para dispositivos eletrônicos.



Fonte: JOGO, 2008

Figura 12. Jogo da memória para crianças

A Figura 12 é um exemplo de jogo da memória, para desktop, voltado para crianças, trabalha com imagens coloridas e alegres, de tamanho médio. Oferece como recurso diferencial a contagem de tentativas do jogador. Ao final do jogo apresenta a quantidade de tentativas e a opção de reiniciar o jogo.



Fonte: MEMÓRIA, 2008

Figura 13. Jogo da memória com relógio

A Figura 13 mostra outro exemplo de jogo da memória voltado para crianças, trabalha com figuras simples, de tamanho mediano que fazem parte do repertório da criança, facilitando a visualização e a memorização. Apresenta um marcador de tempo da duração da partida, ao término deste mostra o valor alcançado e permite reiniciar o jogo.



Fonte: MEMÓRIA, 2008

Figura 14. Jogo da memória de letras e cores

Na Figura 14, há um exemplo de jogo da memória que não visa estimular somente a memória, mas também incentivar a aprendizagem, isto através do uso da combinação de letras e cores. Pode ser considerado com um nível de dificuldade um pouco mais elevado. Marca o tempo do jogo e possui um botão para reiniciar o mesmo. Nenhuma mensagem indicando o final da partida ou a pontuação obtida é exibida.

O modelo exibido na Figura 15 trabalha com imagens simples, de poucas cores e tamanho grande, que facilita a memorização, direcionado para pessoas sem muita prática ou com dificuldade de memorização, também possui um marcador de tempo da partida. No final do jogo, é mostrado o tempo obtido para finalizar a partida e o botão para iniciar uma nova partida.



Fonte: JOGOS, 2008

Figura 15. Jogo da memória para iniciantes

Se quiser recomençar o jogo, é só clicar no botão do tempo.



Fonte: MINGAU, 2008

Figura 16. Jogo da memória dos países

A Figura 16 traz um modelo de jogo da memória bastante didático, que busca estimular o aprendizado através do entretenimento, trabalha com a combinação de imagens e palavras.

Neste jogo, é necessário conhecimento e memorização, pois a formação dos pares é feita através da combinação correta entre o nome do país e sua respectiva bandeira, é um jogo que apresenta um grau de dificuldade maior. Oferece texto de instruções e um marcador de tempo. Também apresenta o tempo de duração da partida no fim do jogo e o botão para começar outro jogo.

2.5 Documentos de Requisitos

2.5.1 Visão Geral do Sistema

O jogo da memória para dispositivos móveis deve oferecer um menu com opções de: ajuda, novo jogo, pausar, sair, continuar, ranking e nível de dificuldade. As partidas são controladas pelo número de jogadas e finalizadas quando todos os pares forem encontrados. O jogo deve oferecer ao jogador a possibilidade de interromper ou finalizar a partida quando desejar, através da opção pausar ou sair. Ao encerrar a partida, são apresentados os resultados obtidos e, caso os valores sejam melhores dos que já estão gravados, o jogador pode incluir seu nome no ranking.

2.5.2 Requisitos Funcionais

- a) O jogo da memória deve possuir uma biblioteca de imagens que são carregadas antes da inicialização do jogo. As imagens têm um nome e são classificadas por categoria de nível;
- b) Cada nível é representado por um nome e pela quantidade de cartas utilizadas no jogo;
- c) O jogo trabalha com os logotipos dos cursos do Unisalesiano e seus respectivos nomes;
- d) A cada nova partida, essas imagens são sorteadas e selecionadas, de acordo com o nível de dificuldade, e atribuídas a um par de cartas. Estas cartas são identificadas pela imagem e valor, que são

utilizados para comparar e identificar os pares, posição X e posição Y na tela do dispositivo móvel, e situação virada ou aberta;

- e) O jogo apresenta uma tela de “abertura”;
- f) O jogo apresenta um menu que oferece as seguintes opções: ajuda, novo jogo, pausar jogo, sair, continuar, ranking, nível de dificuldade;
- g) Antes da partida ser iniciada, as cartas com as imagens voltadas para cima, são apresentadas, por alguns segundos, para que o jogador possa memorizar as posições;
- h) O jogo oferece três níveis de dificuldade, baseados na quantidade de cartas e no conhecimento sobre os logotipos dos cursos do Unisalesiano:
 - nível 1: possui 12 cartas ilustradas somente com figuras das logo;
 - nível 2: possui 20 cartas ilustradas somente com figuras das logo;
 - nível 3: possui 10 cartas ilustradas com figuras das logo e 10 cartas com o nome do curso associado a logo.
- i) O jogo possui um cronômetro para marcar o início e término da partida, cujos valores são utilizados para analisar a classificação do jogador;
- j) O jogo deve disponibilizar um procedimento para contar quantidade de jogadas realizada pelo jogador;
- k) Em cada jogada o jogo deve permitir que o jogador selecione apenas duas cartas;
- l) O jogo oferece um ranking de classificação por nível de dificuldade, onde são armazenados os dados do jogador, que atenda aos quesitos necessários para inclusão no mesmo: nome do jogador, classificação, tempo, número de jogadas;
- m) O jogo é encerrado quando o usuário selecionar a opção sair, no menu ou quando todos os pares das cartas forem encontrados;
- n) Ao final da partida é apresentada uma mensagem com o tempo e o número de jogadas efetuadas;
- o) Após o término da partida, o jogo verifica a quantidade de jogadas efetuadas e tempo jogado. Caso os pontos do usuário sejam menores dos já gravados no ranking, ele pode digitar seu nome para que ficar gravado no ranking de classificação;

- p) O sistema armazena apenas as três melhores posições, aquelas que possuem os menores tempos e as menores quantidades de jogadas;
- q) O jogo pode ser pausado e continuado a qualquer momento, quando o usuário aperta o botão indicado.

2.5.3 Requisitos Não Funcionais

a) Requisitos de Software:

- Plataforma J2ME (*Java 2 Micro Edition*, plataforma Java direcionada a pequenos dispositivos).
- Perfil MIDP (*Mobile Information Device Profiles*, Perfis de Informação de Dispositivos Móveis).
- Configuração CLDC (*Connected Limited Device Configuration*, Configurações para Dispositivos com Conexão Limitada, componente da plataforma J2ME).
- Máquina Virtual KVM (Kilo Virtual Machine, máquina virtual que oferece suporte ao J2ME)

b) Requisitos de Hardware:

- Tela: 180 pixels de largura x 225 pixels de altura.
- Memória: 128 Kbytes de RAM - armazenar a implementação do MIDP.
 - 32 Kbytes - recursos Java.
 - 8 Kbytes - memória volátil para persistência de dados.
- Dispositivo de entrada: teclado com números de 0 a 9, mais um conjunto de setas.

3 ANÁLISE ORIENTADA A OBJETOS

3.1 Lista de Casos de Uso

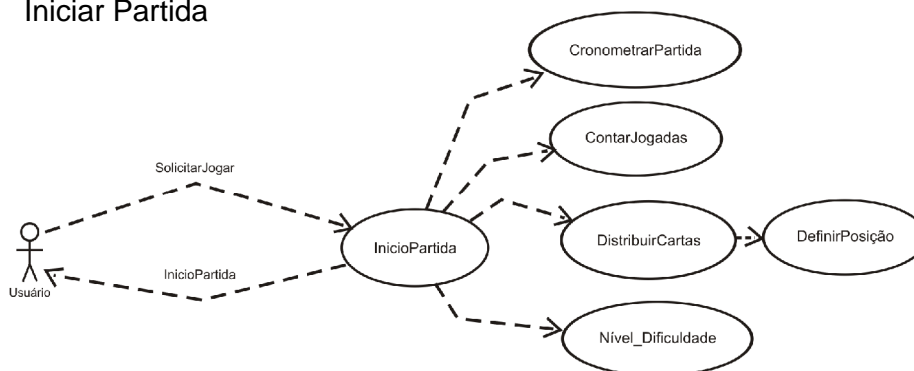
Nº	Descrição do Evento	Evento	Use Case	Resposta
01	Usuário solicita Iniciar partida	DadosPartida	Iniciar partida	Msg01
02	Usuário solicita jogar partida	EscolheCartas	Jogar partida	Msg02
03	Usuário escolhe nível de dificuldade	DadosNívelDificuldade	Escolher nível dificuldade	Msg03
04	Usuário solicita visualizar ranking	DadosRanking	Visualizar ranking	Msg04
05	Usuário solicita pausar partida	SelecionaOpção	Pausar partida	Msg05
06	Usuário solicita reiniciar partida	SelecionaOpção	Reiniciar partida	Msg06
07	Usuário solicita encerrar partida	SelecionaOpção	Encerrar partida	Msg07
08	Usuário solicita sair do jogo	SelecionaOpção	Sair Jogo	Msg08

Fonte: Elaborado pelo autor, 2008

Quadro 1. Lista de Casos de Uso

3.2 Diagrama de Casos de Uso

3.2.1 Iniciar Partida



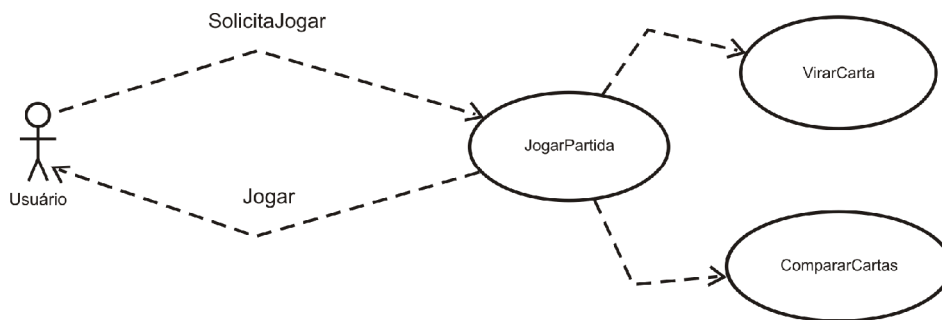
CURSO NORMAL

- 1- O usuário solicita uma nova partida.
- 2- O sistema verifica o nível escolhido, distribui as cartas na tela, primeiro exibe as cartas com as imagens voltadas para cima, após alguns segundos vira as cartas e as exibe com as imagens voltadas para baixo, instancia o cronômetro e o contador de jogadas.

Fonte: Elaborado pelo autor, 2008

Figura 17. Diagrama de Use Case 01 – Iniciar Partida

3.2.2 Jogar Partida



CURSO NORMAL

- 1 - O usuário escolhe no menu a opção jogar.
- 2 - O usuário seleciona a primeira carta e o sistema a exibe na tela.
- 3 - O usuário seleciona a segunda carta e o sistema exibe a mesma na tela.
- 4 - Sistema verifica se as duas cartas selecionadas são iguais.
- 5 - Sistema contabiliza a jogada efetuada.
- 6 - Caso as cartas sejam iguais, permanecerão viradas para cima.
- 7 - O usuário inicia uma nova jogada e repete os passos de 1 a 5
- 8 - Quando todos os pares de cartas são encontrados o sistema encerra a partida.
- 9 - Sistema exibe a pontuação.

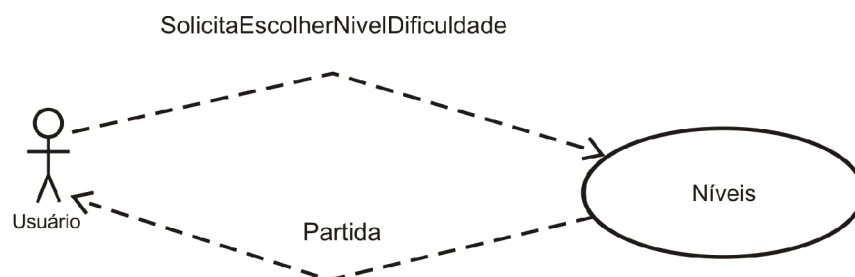
CURSO ALTERNATIVO - CARTAS DIFERENTES

- 5.1 - Caso as cartas sejam diferentes, o sistema volta as mesmas para posição inicial.

Fonte: Elaborado pelo autor, 2008

Figura 18. Diagrama de Use Case 02 – Jogar Partida

3.2.3 Escolher Nível de Dificuldade



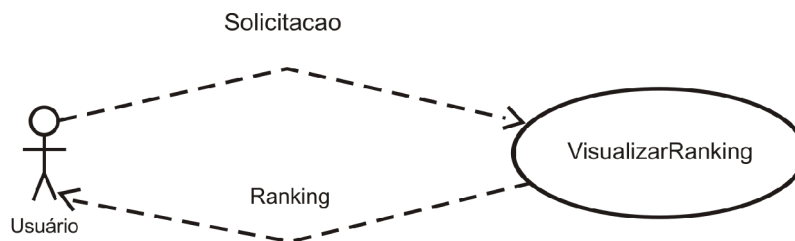
CURSO NORMAL

- 1 - O usuário solicita escolher nível de dificuldade para jogar a partida.
- 2 - O sistema verifica os níveis existentes.
- 3 - O usuário escolhe uma das opções apresentadas.
- 4 - O sistema prepara a próxima partida com o nível de dificuldade escolhido.

Fonte: Elaborado pelo autor, 2008

Figura 19. Diagrama de Use Case 03 – Escolher Nível de Dificuldade

3.2.4 Visualizar Ranking



CURSO NORMAL

- 1- Usuário solicita visualizar do ranking.
- 2- O sistema verifica se existem dados cadastrados no ranking.
- 3- Caso existam dados cadastrados, o sistema exibirá as três primeiras posições, por nível, com seus respectivos nomes, quantidade de jogadas e tempo.

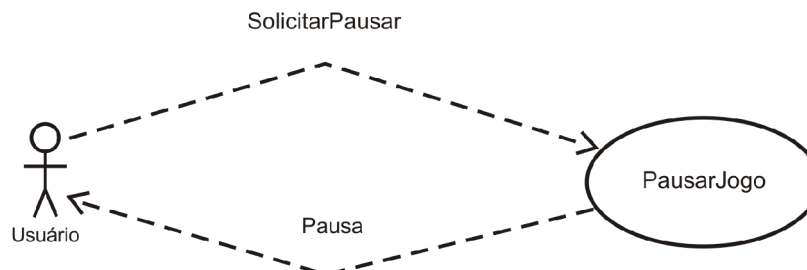
CURSO ALTERNATIVO - CASO NÃO EXISTA CADASTRO NO RANKING

- 2.1 O sistema emite uma mensagem que não há dados cadastrados.
- 2.2 O sistema volta à tela inicial.

Fonte: Elaborado pelo autor, 2008

Figura 20. Diagrama de Use Case 04 – Visualizar Ranking

3.2.5 Pausar Partida



CURSO NORMAL

- 1 - Usuário solicita pausar a partida.
- 2 - O sistema salva os dados da partida jogada até o momento e volta ao menu inicial.

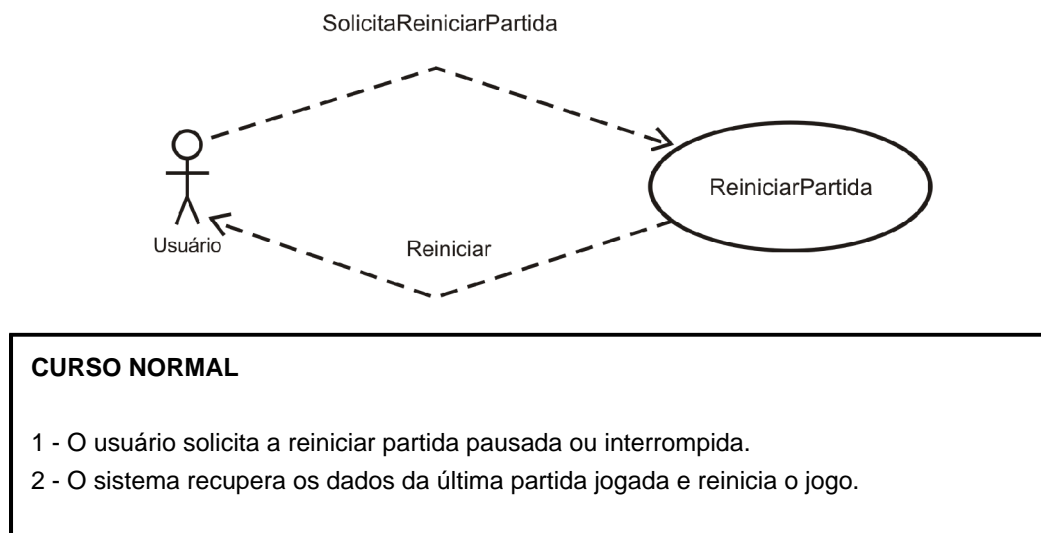
CURSO ALTERNATIVO – INTERRUPTÃO

- 1.1 - Caso ocorra algum imprevisto que force a interrupção.
- 1.2 - O sistema encerra o jogo.

Fonte: Elaborado pelo autor, 2008

Figura 21. Diagrama de Use Case 05 – Pausar Partida

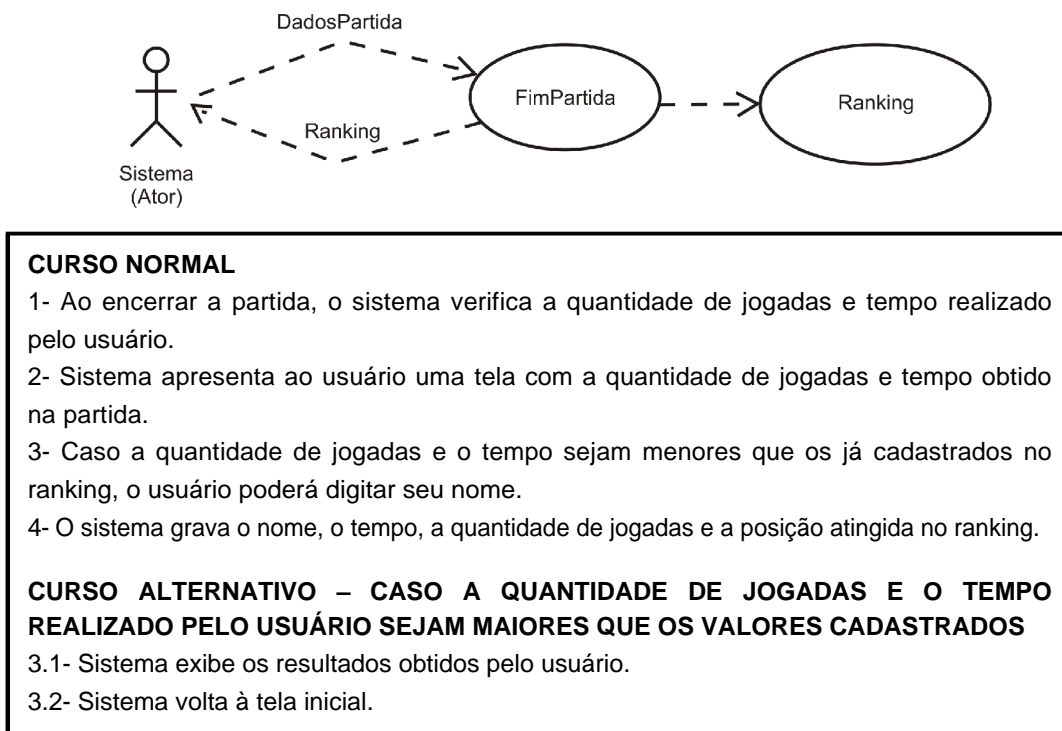
3.2.6 Reiniciar Partida



Fonte: Elaborado pelo autor, 2008

Figura 22. Diagrama de Use Case 06 – Reiniciar Partida

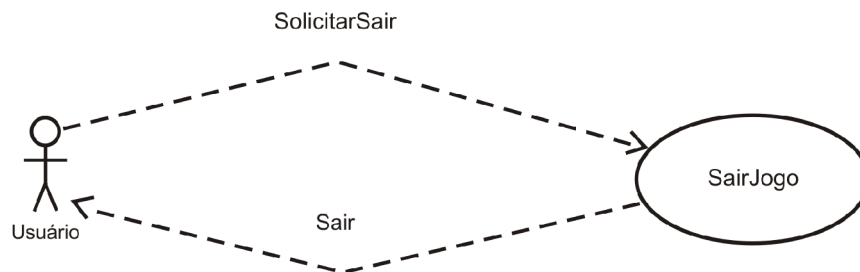
3.2.7 Encerrar Partida



Fonte: Elaborado pelo autor, 2008

Figura 23. Diagrama de Use Case 07 – Encerrar Partida

3.2.8 Sair do Jogo



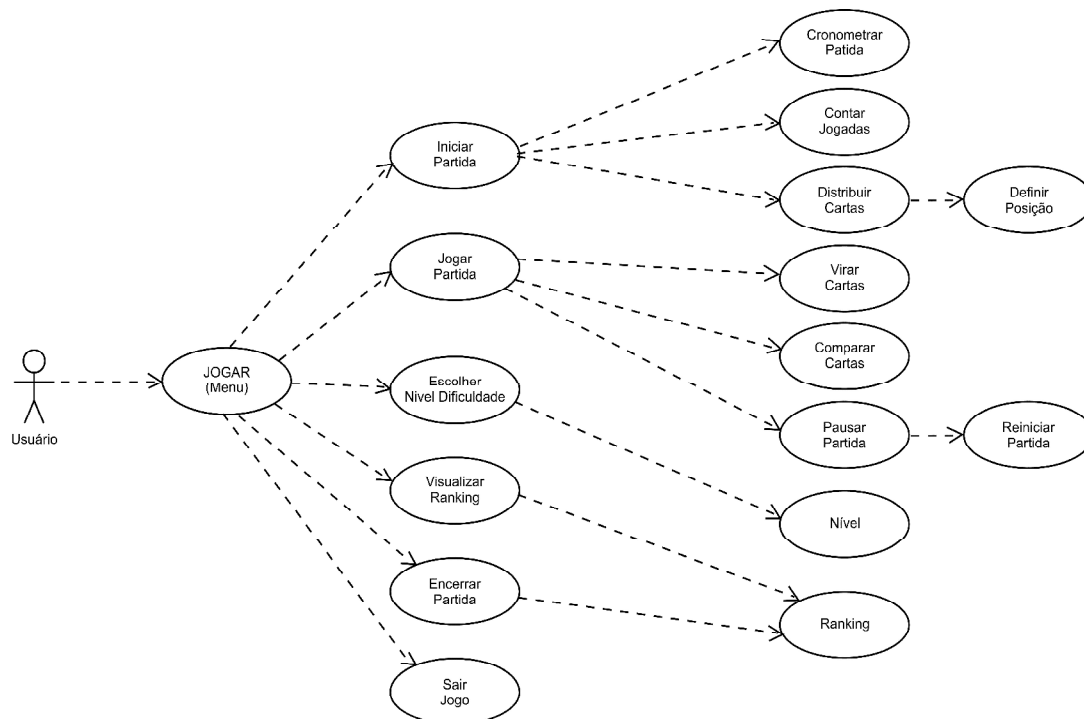
CURSO NORMAL

- 1- Usuário solicita sair do jogo.
- 2- Sistema finaliza o jogo e volta para tela inicial do celular.

Fonte: Elaborado pelo autor, 2008

Figura 24. Diagrama de Use Case 08 – Sair do Jogo

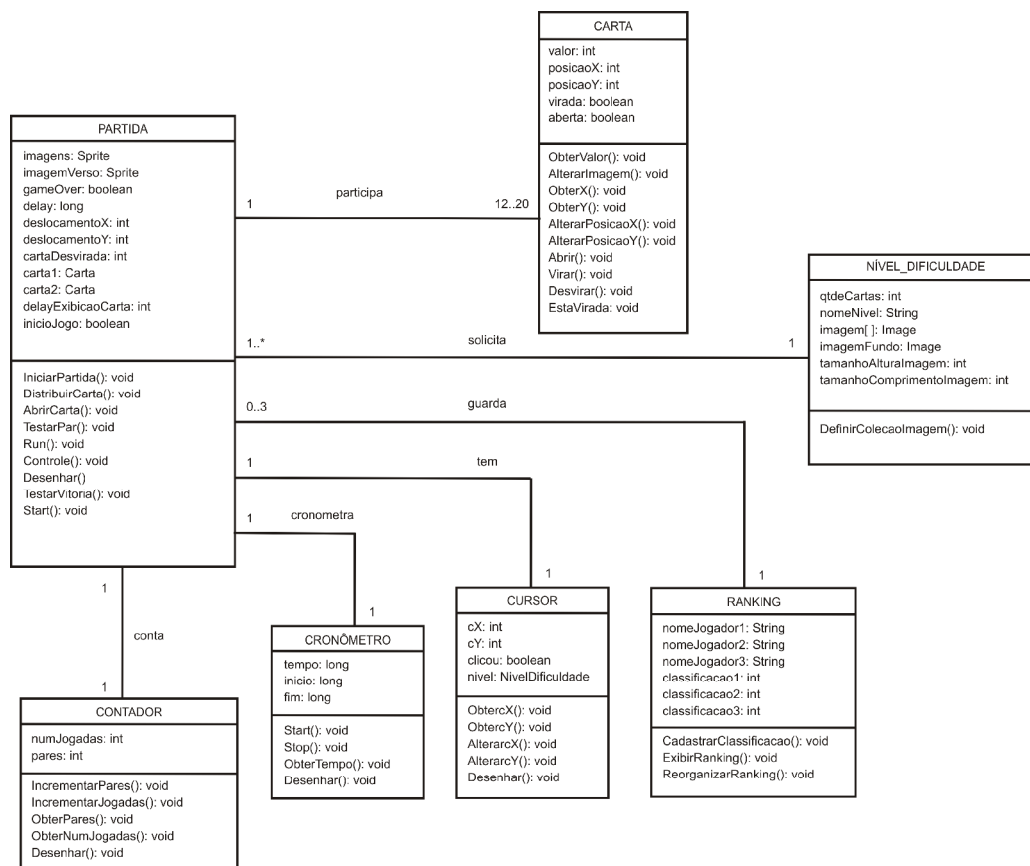
3.2.9 Resumo do Jogo



Fonte: Elaborado pelo autor, 2008

Figura 25. Diagrama de Use Case – Resumo do Jogo

3.3 Diagrama de Classes

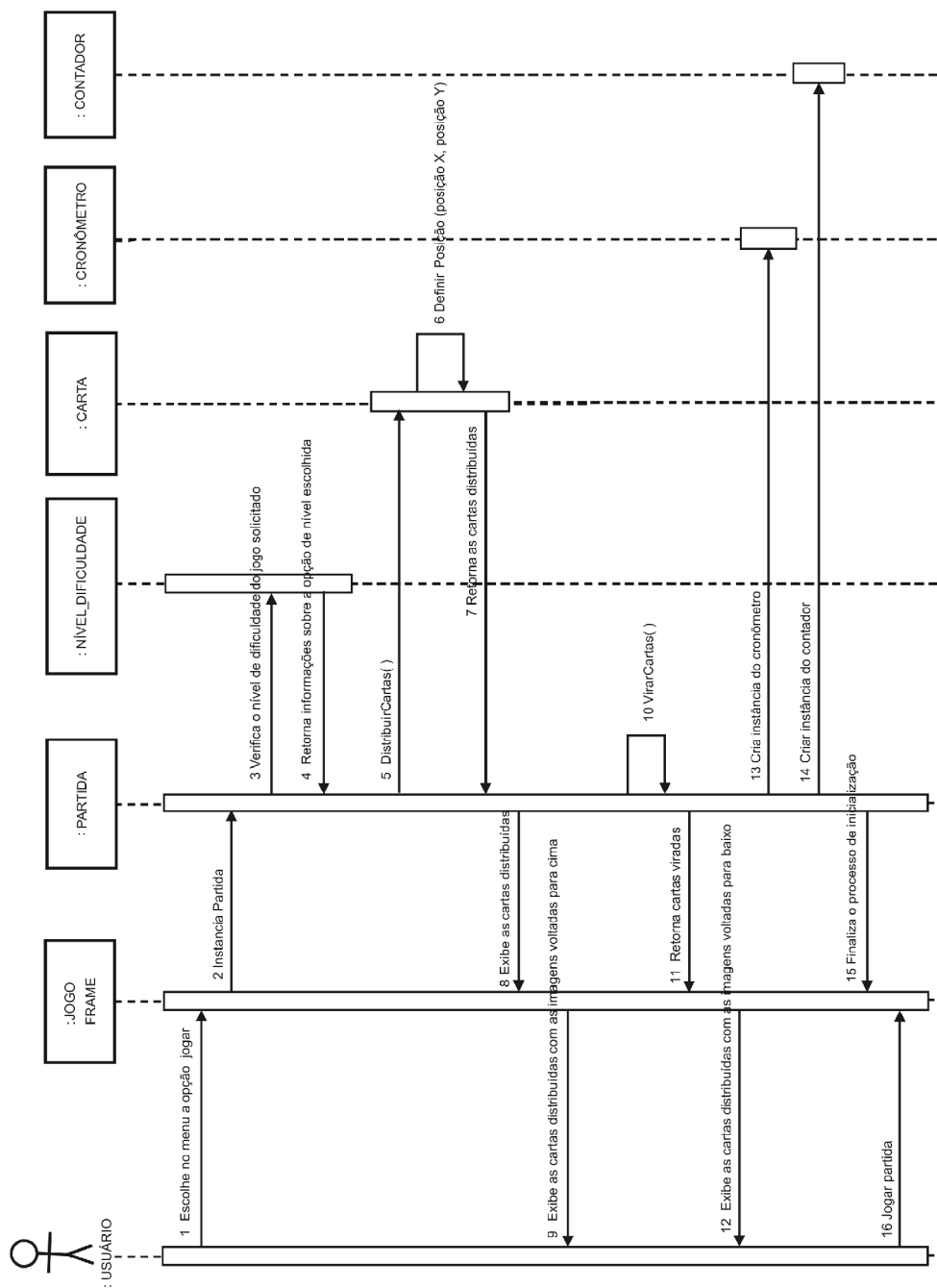


Fonte: Elaborado pelo autor, 2008

Figura 26. Diagrama de Classes

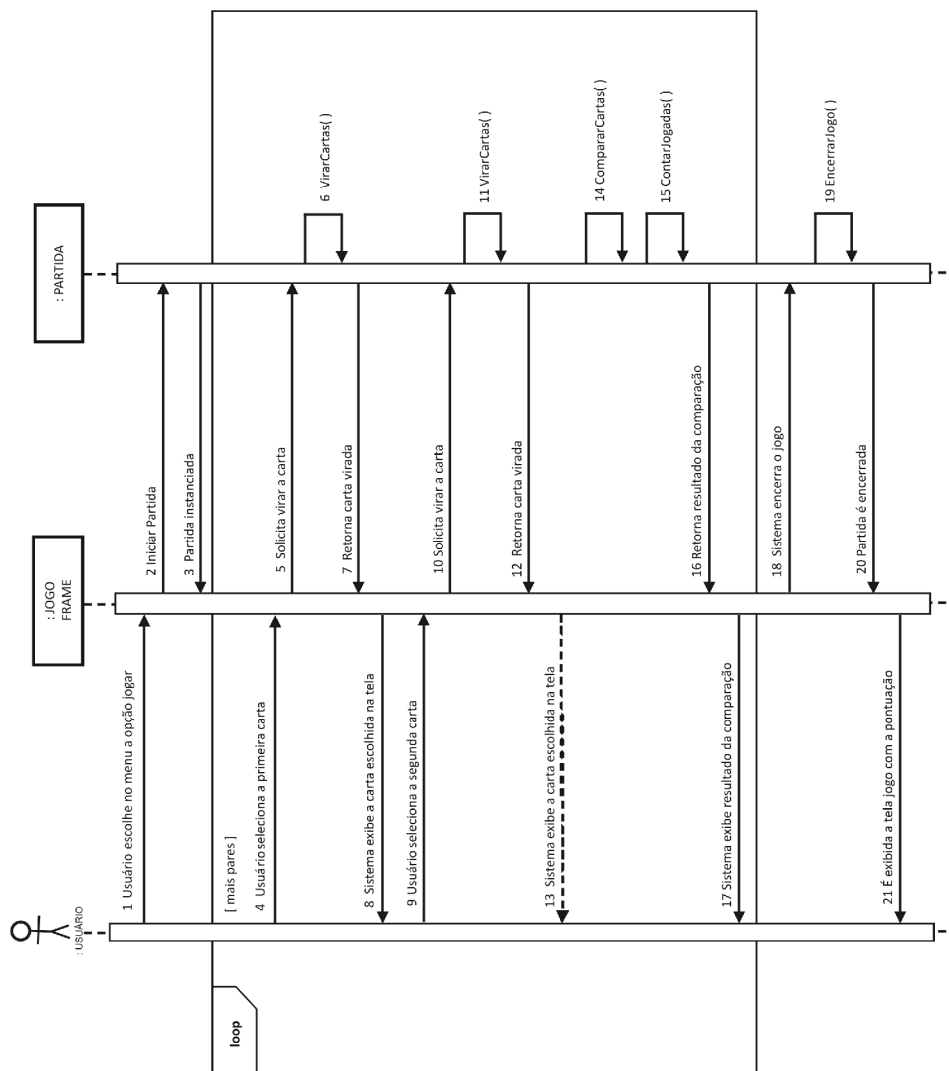
4 PROJETO ORIENTADO A OBJETOS

4.1 Diagramas de Sequência



Fonte: Elaborado pelo autor, 2009

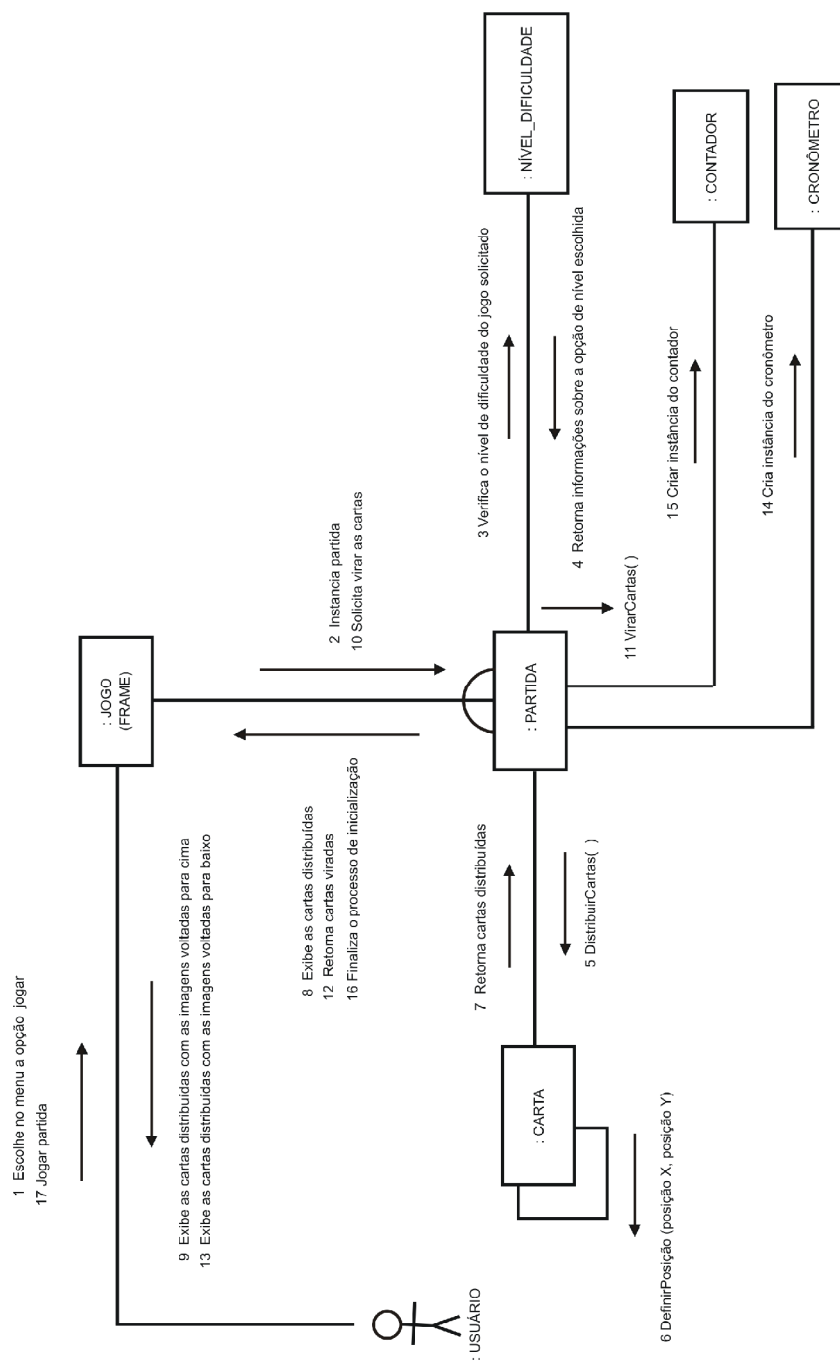
Figura 27. Diagrama de Sequência – Iniciar Partida



Fonte: Elaborado pelo autor, 2009

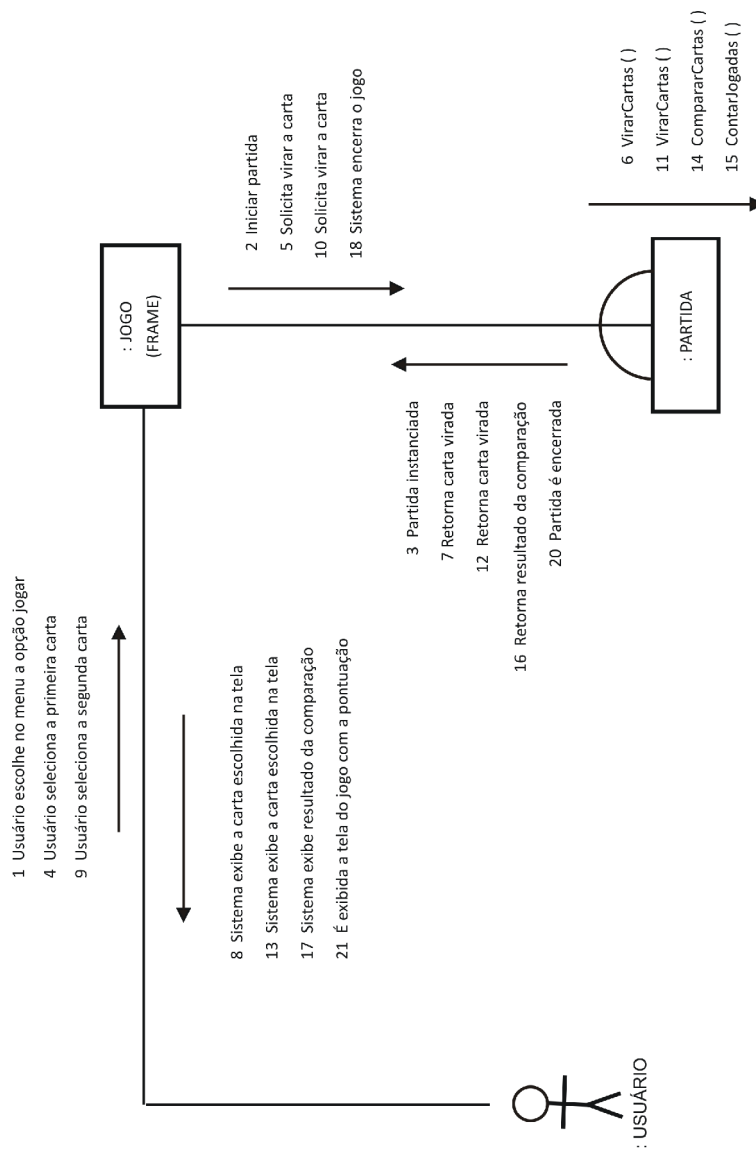
Figura 28. Diagrama de Sequência – Jogar Partida

4.2 Diagrama de Colaboração



Fonte: Elaborado pelo autor, 2009

Figura 29. Diagrama de Colaboração – Iniciar Partida

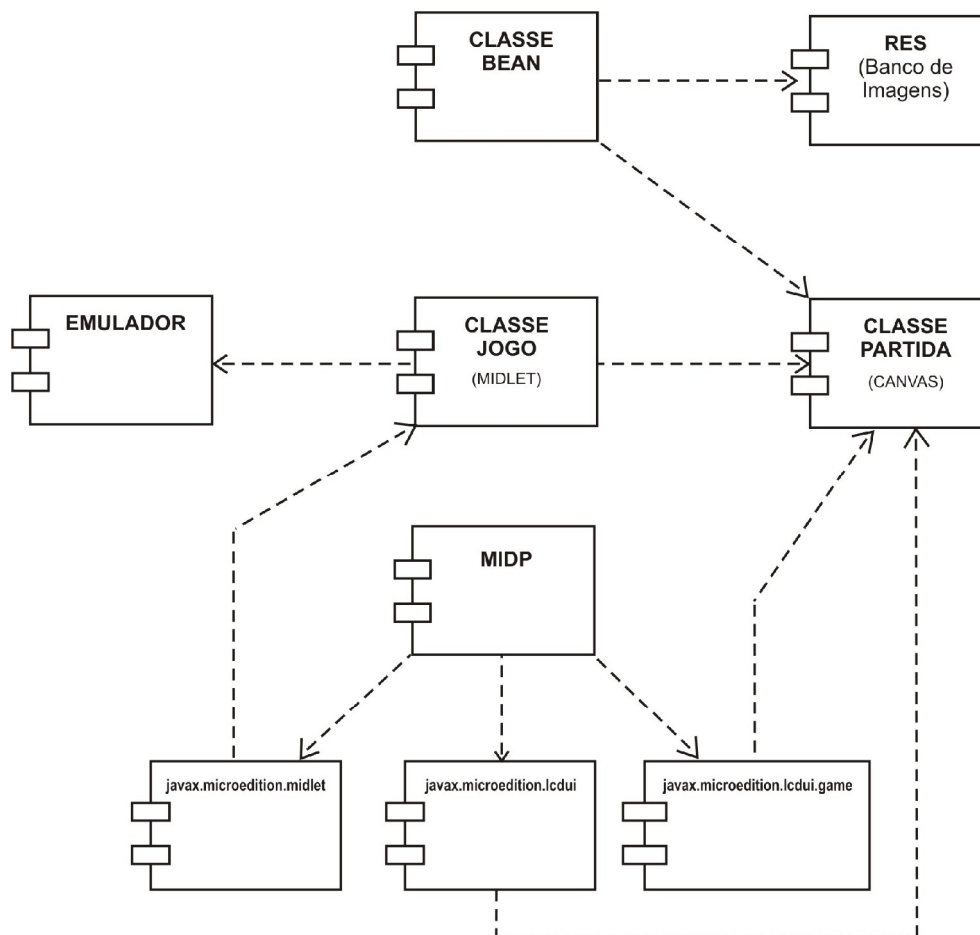


Fonte: Elaborado pelo autor, 2009

Figura 30. Diagrama de Colaboração – Jogar Partida

5 IMPLEMENTAÇÃO ORIENTADA A OBJETOS

5.1 Diagrama de Componentes



Fonte: Elaborado pelo autor, 2009

Figura 31. Diagrama de Componentes

5.2 Layout de Telas

Nesta seção são apresentadas as telas que compõem a aplicação e uma explicação sobre a funcionalidade de cada uma. O detalhamento sobre os recursos utilizados no desenvolvimento das mesmas encontra-se descrito no Anexo 2.



Fonte: Elaborado pelo autor, 2009
 Figura 32. Interface de “abertura”

A Figura 32 representa a interface de “abertura” da aplicação, cuja finalidade é que o jogo não seja iniciado imediatamente, mas somente quando o usuário solicitar uma nova partida. Esta tela oferece também um

menu para escolher uma nova partida, o nível de dificuldade, visualizar o ranking, pausar, continuar e ver ajuda.



Fonte: Elaborado pelo autor, 2009
 Figura 33. Interfaces do início do jogo

A Figura 33 apresenta as interfaces que marcam o início do jogo, são apresentadas após o usuário selecionar no menu a opção “Novo Jogo”.

Inicialmente as cartas são mostradas com as imagens voltadas para cima, por alguns segundos, para que o jogador possa rapidamente memorizar as posições das mesmas.

Em seguida, as cartas são dispostas com o verso voltado para cima,

(com o logotipo do Unisaesiano visível); o marcador do número de jogadas e acertos é apresentado e é iniciado o jogo.

Nestas interfaces são expostas as quantidades de cartas que irão compor o jogo. A quantidade de cartas que será exibida está baseada no nível escolhido, neste exemplo foi escolhido o nível II.



Fonte: Elaborado pelo autor, 2009

Figura 34. Interface do jogo

A Figura 34 reproduz a interface do jogo em andamento, onde ainda não foram encontrados todos os pares. Nela são exibidos tanto os pares encontrados, com as figuras voltadas para cima, quanto as cartas fechadas (sem par formado), com as figuras voltadas para baixo.

Os pares de cartas encontrados são mantidos com as figuras voltadas para cima até a finalização do jogo.

Cada jogada realizada é contabilizada; entende-se jogada a escolha e comparação de duas cartas.

Para apresentação de um layout mais agradável e também facilitar a identificação das cartas fechadas, é utilizada uma imagem única, o logotipo do Unisaesiano, no verso de todas as cartas fechadas.



Fonte: Elaborado pelo autor, 2009

Figura 35. Interface da finalização da partida

A Figura 35 mostra a interface que representa a finalização do jogo, quando todos os pares de cartas são encontrados, e o objetivo principal do jogo é atingido.

As cartas são mostradas na tela com as imagens voltadas para cima.

A seguir outra tela é apresentada com número total de jogadas realizadas durante a partida e o percentual de aproveitamento do jogo.

CONCLUSÃO

O interesse pelo assunto: a tecnologia J2ME, tão em evidência no momento, e o ânimo em desenvolver um projeto que pudesse agregar valor à vida profissional foram fatores que impulsionaram este trabalho.

Assim, o desenvolvimento de um jogo da memória para dispositivos móveis, utilizando a tecnologia J2ME foi uma experiência satisfatória, que permitiu a execução de um projeto diferenciado e o aprendizado da plataforma J2ME, atingindo os objetivos propostos.

Entretanto, a implementação do jogo ocorreu de forma parcial, em razão da limitação de tempo e da complexidade do projeto, sendo desenvolvidas somente as principais funcionalidades do jogo:

- a) a distribuição das cartas;
- b) a execução do jogo (escolha e comparação das cartas);
- c) a contagem das jogadas.

A conclusão do projeto, tendo como resultado final o jogo funcionando, comprova que o desenvolvimento de jogos é uma realidade próxima e viável. Assim, o objetivo deste estudo, de servir como um referencial básico da tecnologia J2ME para os interessados no assunto, foi atingido.

A parte conceitual da tecnologia também foi abordada, visando fornecer maior conhecimento sobre o assunto, e conseqüentemente facilitar a compreensão e aplicação das ferramentas.

O fato de não se limitar a apenas apresentar ferramentas e métodos ou interpretar e desenvolver códigos da tecnologia J2ME foi algo bastante enriquecedor sob o ponto de vista da aprendizagem, uma vez que possibilitou o conhecimento, e também, a preparação e adaptação do ambiente para o desenvolvimento das aplicações.

A *IDE Eclipse Europa*; o *Sun Java Wireless Toolkit* e o *Plugin Eclipse ME* foram as ferramentas instaladas e configuradas, para tornar o ambiente favorável ao desenvolvimento, fornecendo elementos facilitadores.

Este estudo demonstrou ainda que as tecnologias estão disponíveis, oferecendo seus recursos, possibilidades e ferramentas; entretanto, as soluções partem do bom uso que se faz das mesmas e da criatividade do

programador. Toda tecnologia deve ser explorada, experimentada e compartilhada, visando obter resultados inovadores e eficientes.

Segundo Taurion (2009), o que surge como forte tendência é o uso de *OpenSource* para sistemas embarcados, trazendo como consequência a redução de custos e possivelmente a padronização dos celulares. O assunto em questão é amplo e atual, possibilitando diversos caminhos a serem percorridos e infinitas contribuições ainda estão por vir.

REFERÊNCIAS

ALECRIM, E. Máquina Virtual Java (Java Virtual Machine). 22 ago. 2005.

Infowester. 2005. Disponível em: <<http://www.infowester.com/jvm.php>>.

Acesso em 30 set. 2008.

AS NOVIDADES do mundo Java diretamente da JavaOne 2008. **Mundo Java**.

Curitiba: Mundo, n. 30, p. 12-18, ago. 2008.

BARBOZA, D. C.; SILVA, J. C. DA. **Fundamentos do desenvolvimento de jogos para dispositivos móveis e sua aplicação na publicidade**. UNIFESO, Teresópolis, 2007.

BICALHO, R. Documentar Projetos de TI: Fardo ou Necessidade? 21 jul. 2008.

Meio Bit. 2008. Disponível em: <<http://meiobit.pop.com.br/documentar-projetos-de-ti-fardo-ou-necessidade>>. Acesso em 25 set. 2008.

CORASSA, R. Dicas e tendências para games móveis. 2008. **Uol**. Disponível em: <<http://idgnow.uol.com.br/chat/2008/02/29/dicas-e-tendencias-para-games-moveis/>>. Acesso em 29 out. 2008.

DESENVOLVIMENTO de jogos digitais para celulares. **Game Reporter**. 2006.

Disponível em: <<http://www.gamereporter.org/2006/12/18/desenvolvimento-de-jogos-digitais-para-celulares>>. Acesso em 29 out. 2008.

ECLIPSE Downloads. **Eclipse**. 2009. Disponível em:

<<http://www.eclipse.org/downloads/>>. Acesso em 27 fev. 2009.

ECLIPSE ME. **Eclipse ME**. 2009. Disponível em:

<<http://www.eclipse.org/updates/>>. Acesso em 27 fev. 2009.

ESMIN, A. A. A. Modelando com UML – Unified Modeling Language. 2008.

Disponível em: <<http://www.dcc.ufla.br/infocomp/artigos/v1.1/tutorialUML.pdf>>.

Acesso em 15 set. 2008.

FAGUNDES, R. G. L. Criando jogos para celular a partir do estudo de jogos antigos. 2008. **Devmedia** Disponível em:

<http://www.devmedia.com.br/articles/viewcomp.asp?comp=8327&hl=*jogo>.

Acesso em 29 out. 2008.

FONSECA, E. Móble Java ou Java ME. **Java**. 2005. Disponível em:

<http://www.java.com/pt_BR/download/faq/whatis_j2me.xml>. Acesso em: 10 jul. 2008.

FOWLER, M. **UML essencial**: um breve guia para a linguagem padrão de modelagem de objetos. 3.ed. Porto Alegre, Bookman, 2005.

FREITAS, M. Fundamentos de UML. **Devmedia**. 2008. Disponível em:
<<http://www.devmedia.com.br/articles/viewcomp.asp?comp=8640>>. Acesso em 25 set. 2008.

JAVA (linguagem de programação). **Wikipédia**. 2008. Disponível em:
<[http://pt.wikipedia.org/wiki/Java_\(linguagem_de_programa%C3%A7%C3%A3o\)](http://pt.wikipedia.org/wiki/Java_(linguagem_de_programa%C3%A7%C3%A3o))>. Acesso em 15 set. 2008.

JAVA ME Downloads. **Sun Microsystems**. 2009. Disponível em:
<<http://java.sun.com/javame/downloads>>. Acesso em 27 fev. 2009.

JOGO da memória. **Wikipédia**. 2008. Disponível em:
<http://pt.wikipedia.org/wiki/Jogo_da_mem%C3%B3ria>. Acesso em 20 out. 2008.

JOGO da memória. 2008. **Cabovisão**. Disponível em:
<http://clientes.netvisao.pt/mcharrao/sala_de_estudo_virtual/memoria.htm>. Acesso em 30 set. 2008.

JOGO de computador. **Wikipédia**. 2008. Disponível em:
<http://pt.wikipedia.org/wiki/Jogo_de_computador>. Acesso em 20 out. 2008.

JOGOS. 2008. **Site de Dicas Uol**. Disponível em:
<http://sitededicadas.uol.com.br/jogos/jogo_memoria.htm>. Acesso em 30 set. 2008.

JOGOS para celular. 2005. **UCEL**. Disponível em:
<<http://www.ucel.com.br/servico/serv08.asp>>. Acesso em 20 out. 2008.

JOGOS para celulares – Aprenda a desenvolver o seu. 2008. **Dukemidp Código Livre**. Disponível em:
<<http://dukemidp.codigolivre.org.br/projetos.htm>>. Acesso em 30 set. 2008.

JOHNSON, T. M. **Java para dispositivos móveis**. Desenvolvendo aplicações com J2ME. São Paulo: Novatec, 2008.

LARMAN, C. **Utilizando UML e padrões**. 3.ed. Porto Alegre: Bookman, 2007.

LEITE, M. Linguagem de programação: qual a melhor? 06 abr. 2006. **TI Master**. 2006. Disponível em
<http://www.timaster.com.br/revista/artigos/main_artigo.asp?codigo=1108_paq=2>. Acesso em 30 set. 2008.

MATTOS, R. Linguagem de Programação. 18 out. 2004. **Linha de Código**. 2004. Disponível em: <<http://www.linhadecodigo.com.br/Artigo.aspx?id=489&pag=1>>. Acesso em 25 set. 2008.

MEMÓRIA. 2008. **Alzira Zulmira**. Disponível em:
<<http://www.alzirazulmira.com/concentration/memoria.htm>>. Acesso em 30 set. 2008.

MEMÓRIA. 2008. **Divertudo**. Disponível em:
<<http://www.divertudo.com.br/memoria/memoria.html>>. Acesso em 30 set. 2008.

MIDP. **Wikipédia**. 2008. Disponível em: <<http://www.pt.wikipedia.org/wiki/midp>>. Acesso em 10 set. 2008.

MINGAU Digital. 2008. **Mingau Digital**. Disponível em:
<http://www.mingaudigital.com.br/article.php3?id_article=401>. Acesso em 30 set. 2008.

MUCHOW, J. W. **Core J2ME**. Tecnologia & Midp. São Paulo: Makron Books do Brasil, 2006.

NOGUEIRA, A. Histórico da UML 21 fev. 2005. **Imaster**. 2005. Disponível em:
<http://imasters.uol.com.br/artigo/2994/uml/historico_da_uml/>. Acesso em 25 set. 2008.

PAMPLONA, V. F. ago. 2008. **Java Free**. 2008. Disponível em:
<<http://www.javafree.org/content/view.jf?idContent=84>>. Acesso em 25 set. 2008.

RADTKE, P. V. W. Programação Gráfica – Parte 3 Versão em Java. **ppgia**. 2006. Disponível em: < <http://www.ppgia.pucpr.br/~radtke/jogos/seminarios/>>. Acesso em 20 out. 2008.

SANTOS, A. M. C. Desenvolvimento de jogos para dispositivos móveis utilizando a plataforma JAVA ME MIDP 2.0 GAMR API: desafios e oportunidades. 2006. Manaus. **Instituto de Ensino Superior FUCAPI – CESF – Engenharia de Comunicação**. Disponível em:
<<http://www.cesf.br/arquivos/biblioteca/ecm/alexandremagno.pdf>>. Acesso em 20 out. 2008.

SCOTT, K. **O processo unificado explicado**. Porto Alegre: Bookman, 2003.

SILVEIRA, I. F. Linguagem Java. 30 jun. 2003. **Infowester**. 2003. Disponível em: <<http://www.infowester.com/lingjava.php>>. Acesso em 30 set. 2008.

SPOSITO, R. Aplicações móveis devem crescer 102% ao ano. 29 out. 2007. **Info**. 2007. Disponível em:
< <http://info.abril.uol.com.br/aberto/infonews/102007/29102007-13.shl> >. Acesso em 30 set. 2008.

TAURION, C. Java e computação embarcada. Mundo Java. Curitiba: **Mundo Java**. n. 30, p. 74, ago. 2008.

ANEXOS

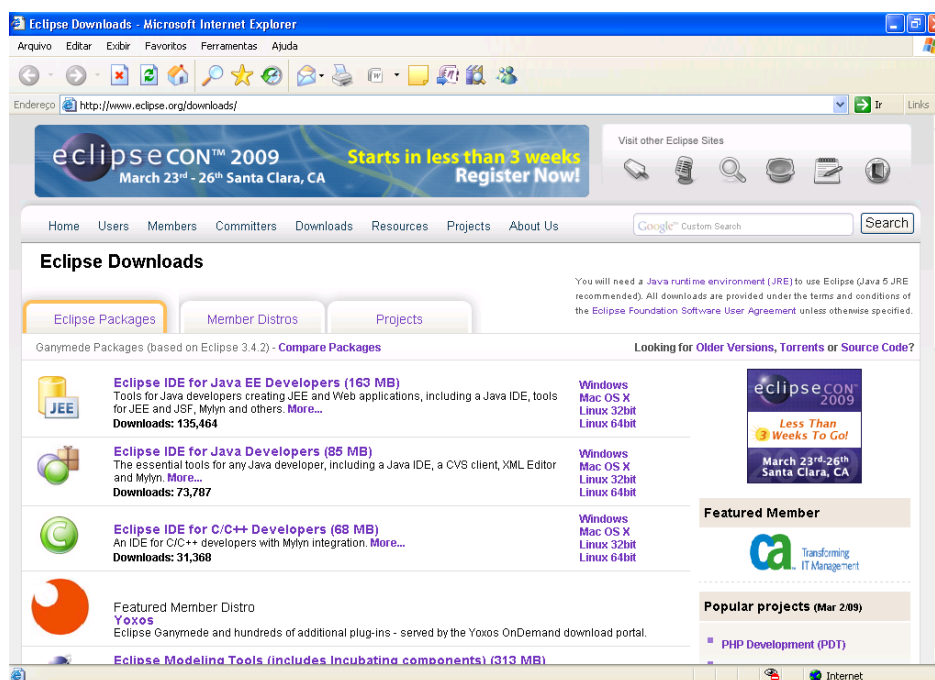
ANEXO A - CONFIGURAÇÃO DO AMBIENTE ECLIPSE PARA O DESENVOLVIMENTO COM J2ME

Para o desenvolvimento de aplicações voltadas para dispositivos móveis, faz-se necessária a instalação de alguns softwares e *plugins*. Neste projeto, as ferramentas adotadas foram as seguintes:

- a) IDE Eclipse Europa (Eclipse Downloads, 2009);
- b) *Sun Java Wireless Toolkit* (Java ME Downloads, 2009);
- c) *Plugin Eclipse ME* (Eclipse ME, 2009).

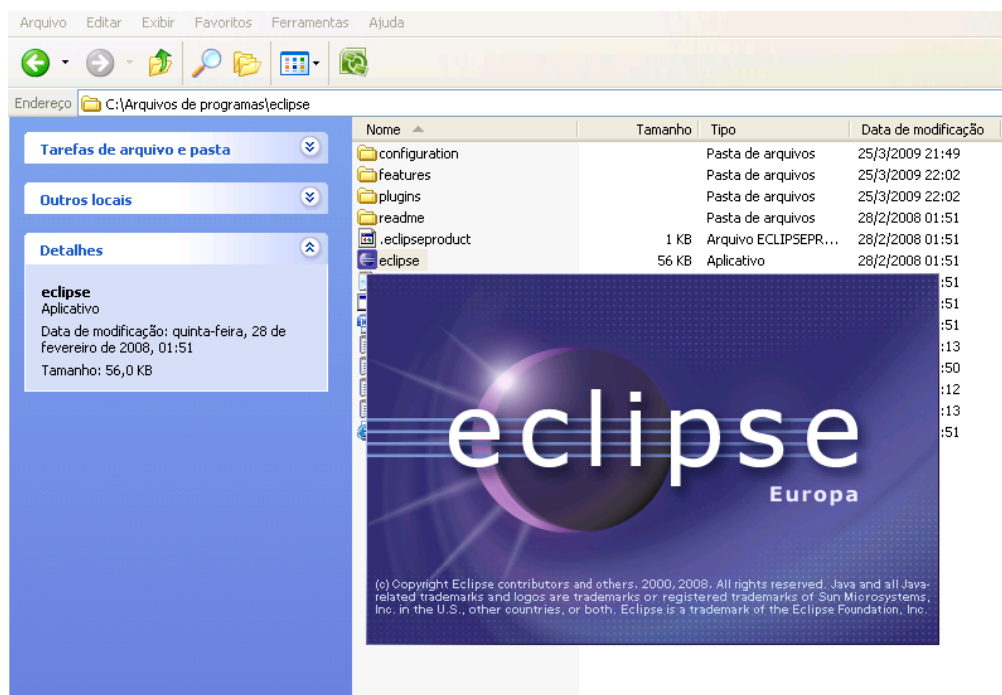
1 PROCESSO DE INSTALAÇÃO DO ECLIPSE

Através do site <http://www.eclipse.org/downloads/> é possível fazer o download da IDE Eclipse, atentando-se para escolher o Sistema Operacional adequado. Em seguida, basta descompactar o arquivo baixado, não há necessidade de instalação.

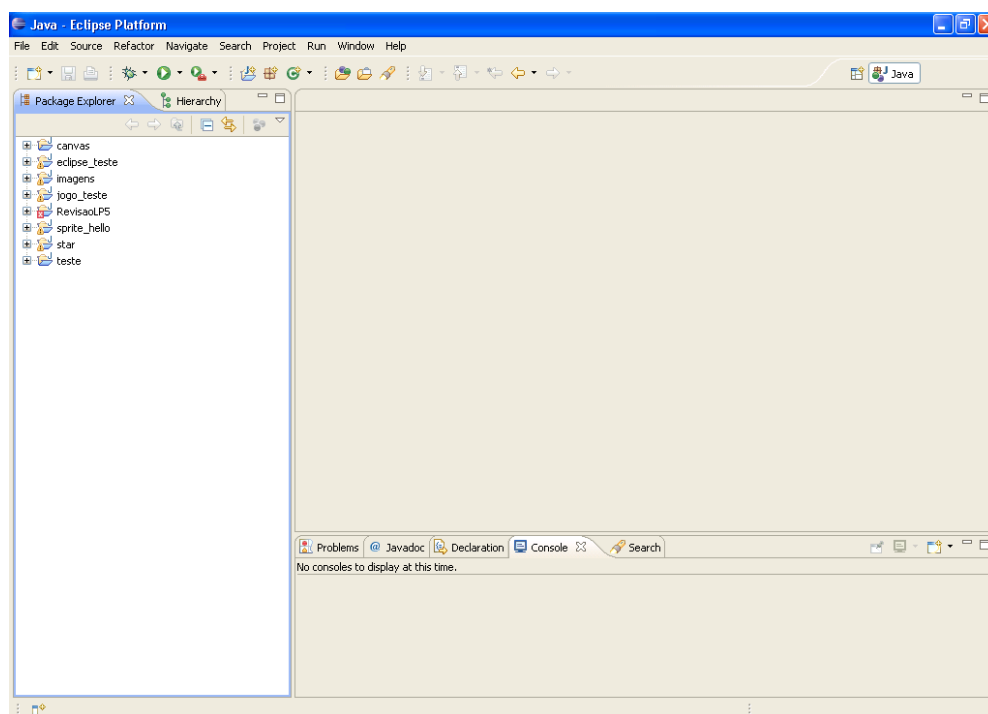


Fonte: Elaborado pelo autor, 2009

Figura 36. Página disponível para o *download* da IDE Eclipse



Fonte: Elaborado pelo autor, 2009
 Figura 37. Execução da IDE Eclipse



Fonte: Elaborado pelo autor, 2009
 Figura 38. Ambiente Eclipse

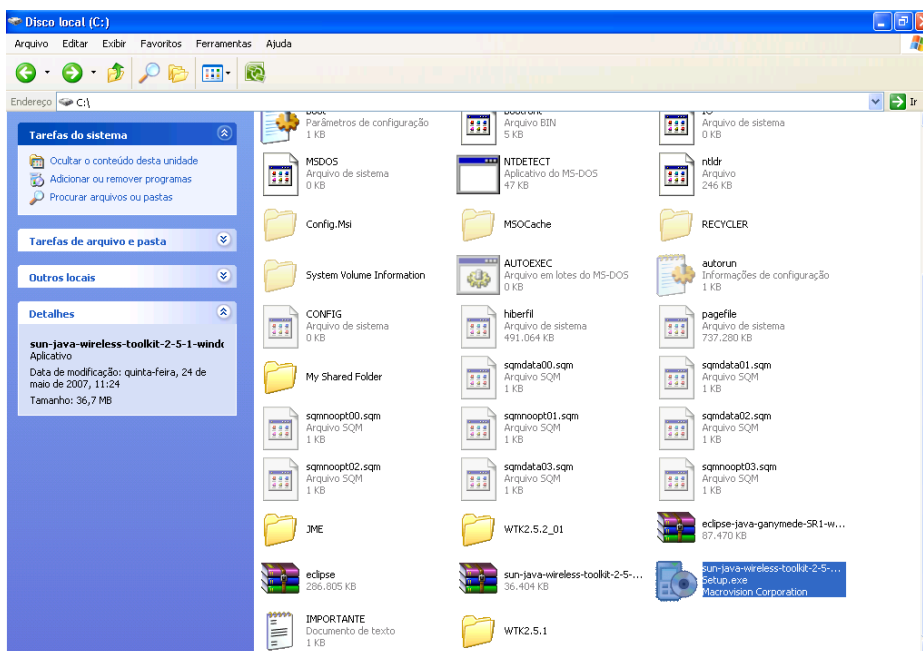
2 PROCESSO DE INSTALAÇÃO DO *SUN WIRELESS TOOLKIT*

Para instalação do *Sun Wireless Toolkit*, é necessário fazer o download do mesmo através do site <http://java.sun.com/javame/downloads>.



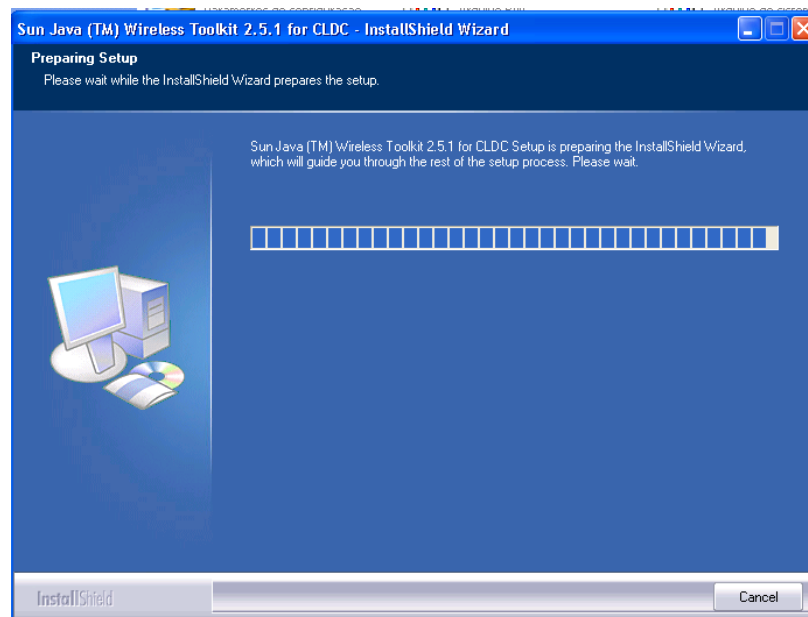
Fonte: Elaborado pelo autor, 2009

Figura 39. Página disponível para o download do *Sun Wireless Toolkit*



Fonte: Elaborado pelo autor, 2009

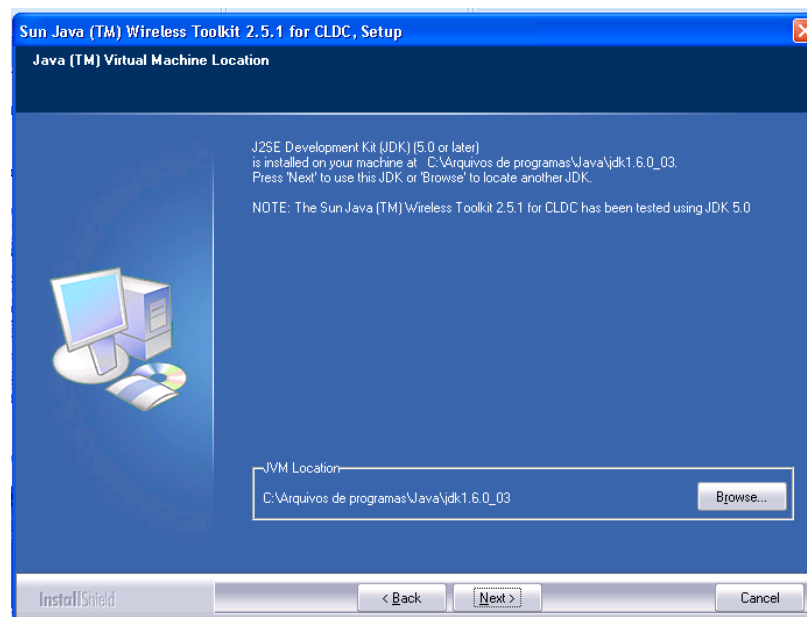
Figura 40. Início da instalação do *Sun Wireless Toolkit*



Fonte: Elaborado pelo autor, 2009

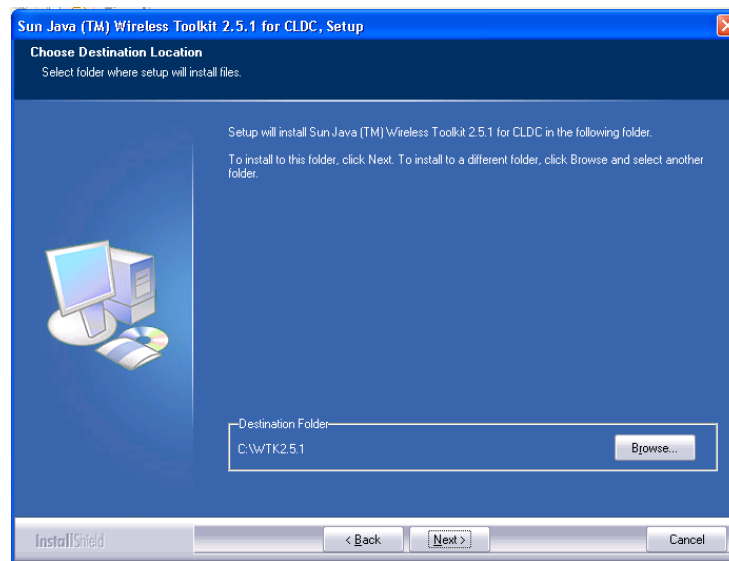
Figura 41. Início da instalação do *Sun Wireless Toolkit*

Durante o processo de instalação, a localização da JVM (Java Virtual Machine) e o local onde será instalado o WTK deve ser informado, normalmente, ambos no C:. Após a instalação, o software estará pronto para ser executado.



Fonte: Elaborado pelo autor, 2009

Figura 42. Tela para configurar a localização da JVM



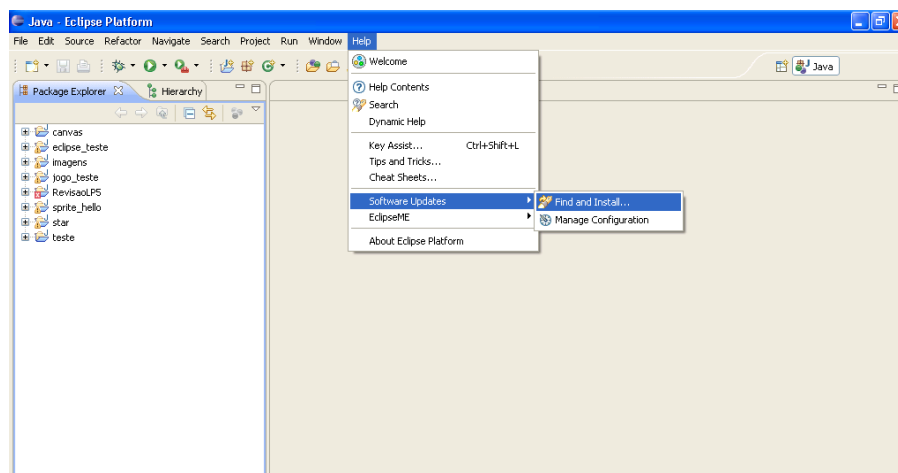
Fonte: Elaborado pelo autor, 2009

Figura 43. Tela para configurar a localização da instalação do WTK

3 PROCESSO DE INSTALAÇÃO DO *PLUGIN* ECLIPSE ME

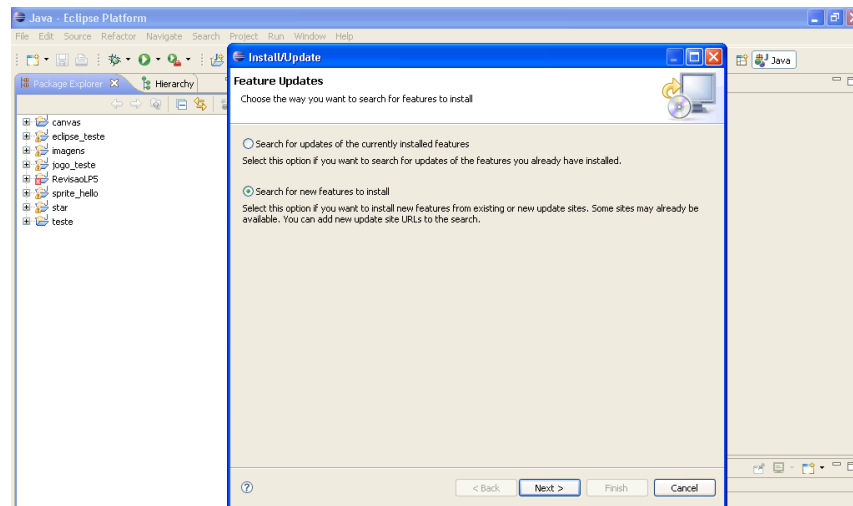
Para o desenvolvimento de aplicações J2ME no ambiente eclipse, é necessária a configuração do *plugin* Eclipse ME, que possibilita a ligação entre a IDE e o Java Wireless Toolkit fornecido pela SUN.

Dentro do ambiente Eclipse, escolha a opção de menu *Help* > *Software Updates* > *Find and Install*. Uma tela para a escolha do tipo de instalação será apresentada. Escolha a opção – *search for new features to install* – que possibilita acessar os repositórios manualmente.



Fonte: Elaborado pelo autor, 2009

Figura 44. Início da instalação do *plugins* Eclipse ME

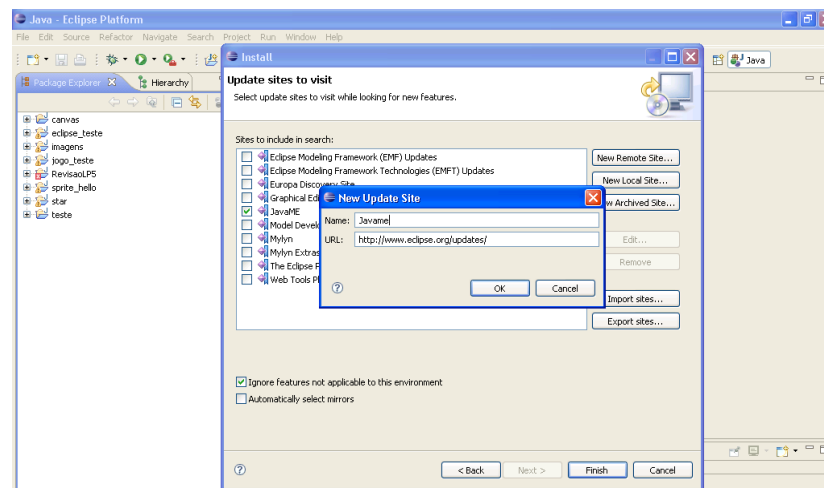


Fonte: Elaborado pelo autor, 2009

Figura 45. Tela para escolher o tipo de instalação

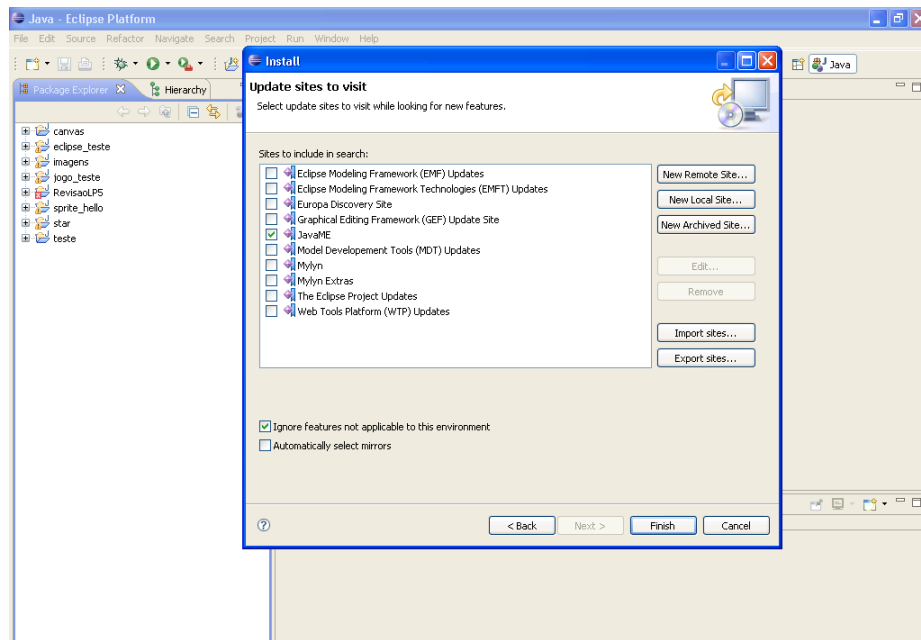
Na próxima tela, escolha a opção *New Remote Site*; na sequência é apresentada a tela *New Update Site* onde devem ser informados o nome do repositório e da url do link do *plugin*, JavaMe e <http://www.eclipse.org/updates/>, respectivamente.

O site JavaMe é adicionado e marcado na lista de sites para busca; esta fase é encerrada com a execução da busca dos *plugins* e a apresentação dos resultados obtidos. Os termos de licença são apresentados e devem ser aceitos, por fim encerra-se a instalação. A seguir os *downloads* serão executados e o Eclipse deve ser reiniciado.



Fonte: Elaborado pelo autor, 2009

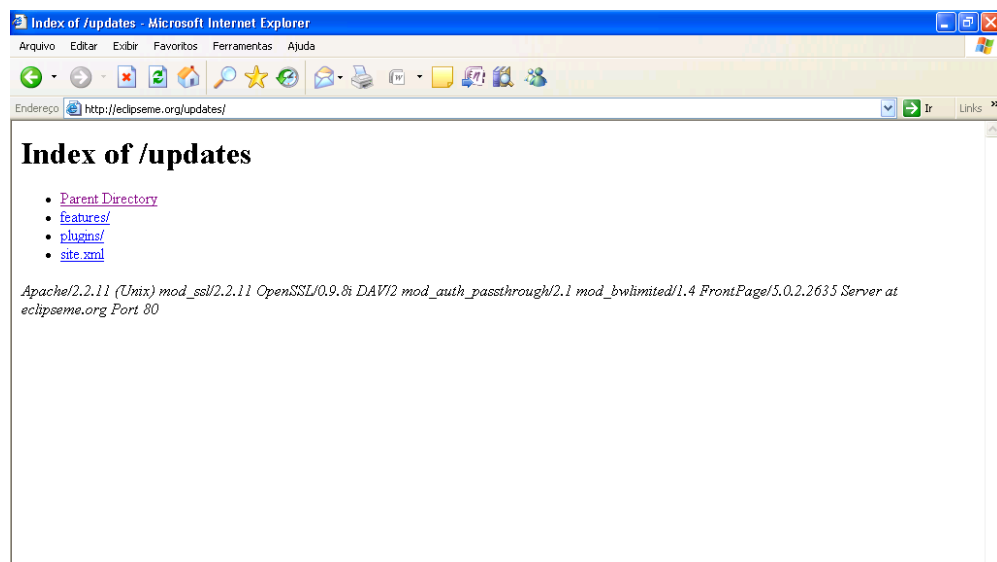
Figura 46. Tela para definir o nome do repositório e da url



Fonte: Elaborado pelo autor, 2009

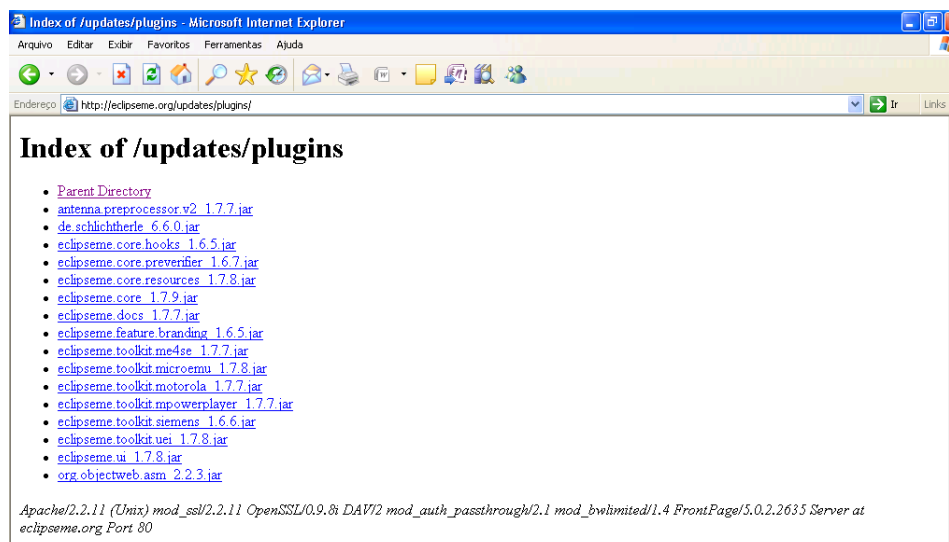
Figura 47. Tela para finalização e instalação dos *plugins*

Este processo tem como finalidade atualizar ou instalar atualizações de *plugins*, entretanto, caso ocorra erros durante este processo, é possível realizar o mesmo de forma manual, acessando o site <http://eclipse.org/updates/>, fazendo o download dos *plugins* e instalando-os na pasta dos *plugins* do Eclipse.



Fonte: Elaborado pelo autor, 2009

Figura 48. Página disponível para fazer o *download* dos *plugins*



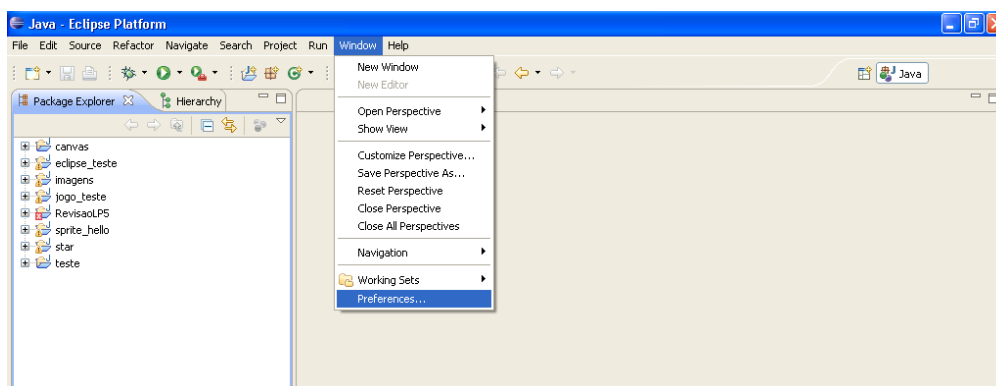
Fonte: Elaborado pelo autor, 2009

Figura 49. Página disponível para fazer o *download* dos *plugins*

4 PROCESSO DE CONFIGURAÇÃO DO ECLIPSE

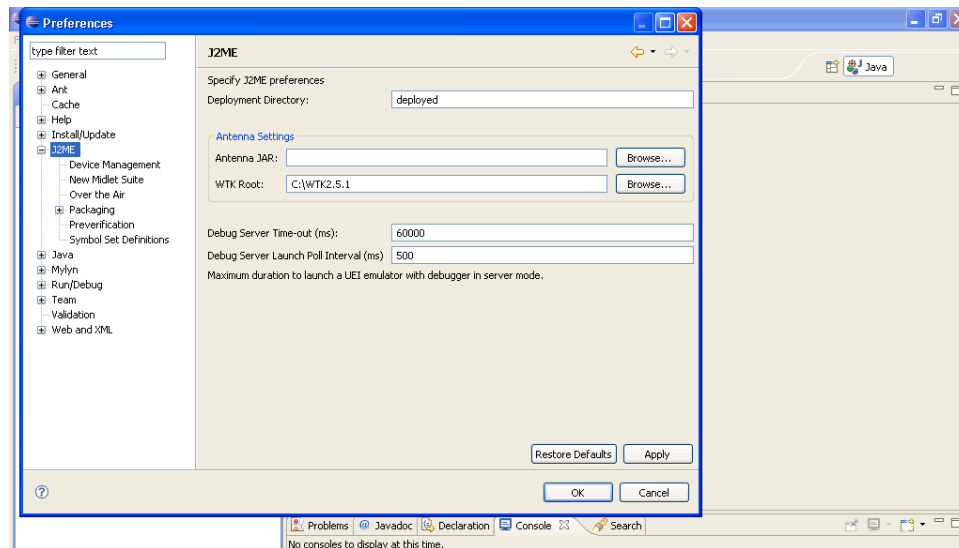
Antes de iniciar o desenvolvimento das aplicações, é necessário realizar algumas configurações no ambiente.

No menu *Window > Preferences* é apresentada uma tela de configurações, onde a opção J2ME deve estar selecionada. Nesta tela, no campo WTK root deve ser informado o nome do local onde está instalado o WTK na máquina (C:\WTK2.5.1), para que seja possível a compilação das aplicações JME. Ainda em *Preferences*, através da opção *Device Management*, deve ser feito o importe dos emuladores que estão na WTK.



Fonte: Elaborado pelo autor, 2009

Figura 50. Acesso a tela de configurações



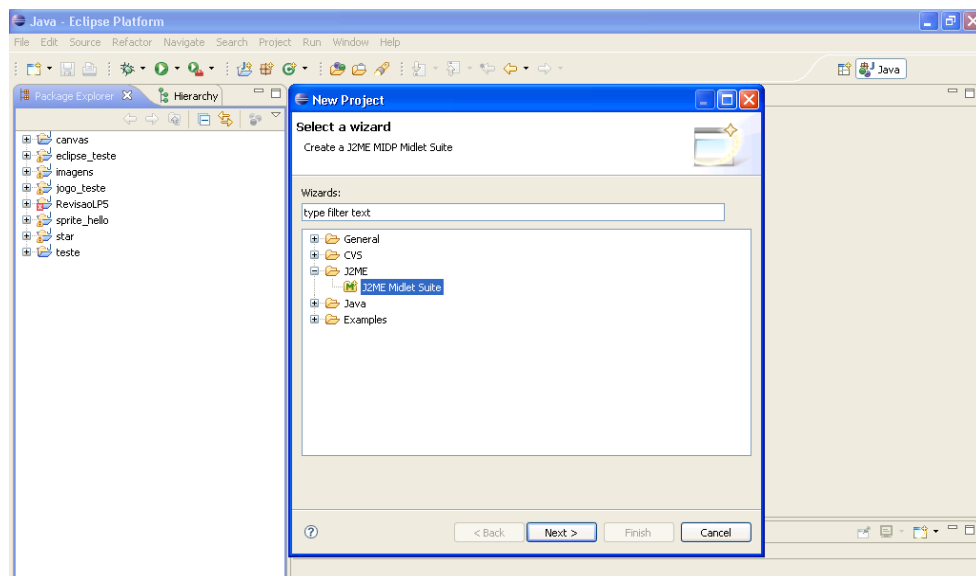
Fonte: Elaborado pelo autor, 2009

Figura 51. Tela de configurações

5 PRIMEIRO PROJETO JME

Para testar a funcionalidade dos softwares e *plugins* instalados, um pequeno projeto deve ser criado e executado.

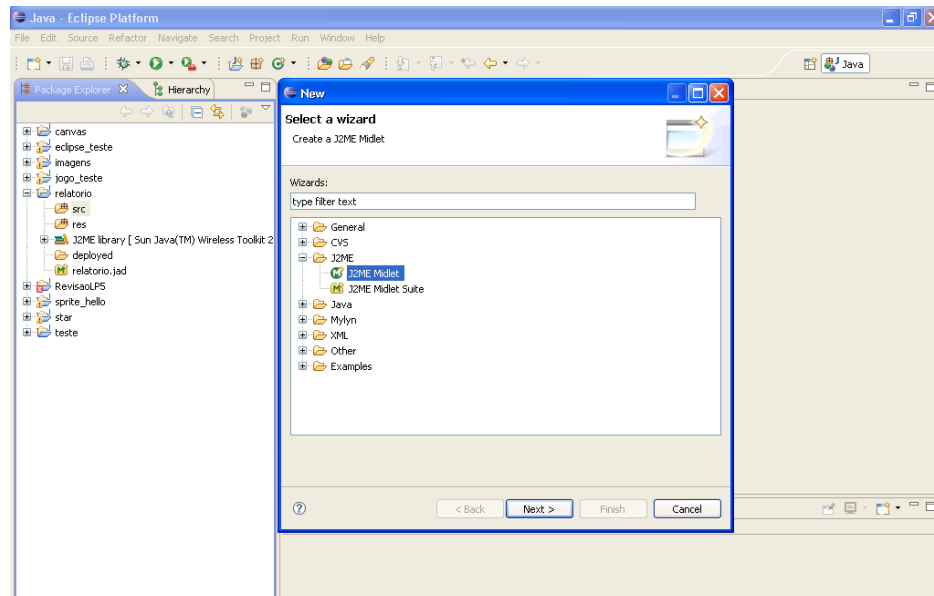
No ambiente Eclipse crie um novo projeto, escolhendo a opção J2ME *Midlet Suite*. Neste processo, toda estrutura do projeto é criada.



Fonte: Elaborado pelo autor, 2009

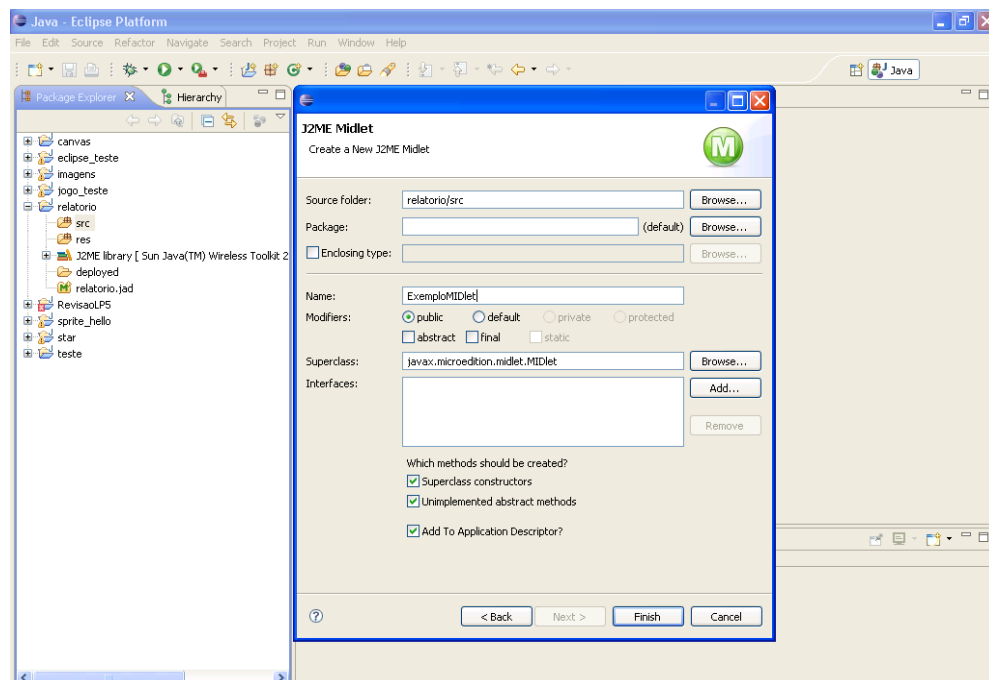
Figura 52. Tela para criação do novo projeto

A seguir crie uma *Midlet*, dentro do pacote *src*;;, nesta operação o construtor e os métodos básicos da *Midlet* são criados automaticamente.



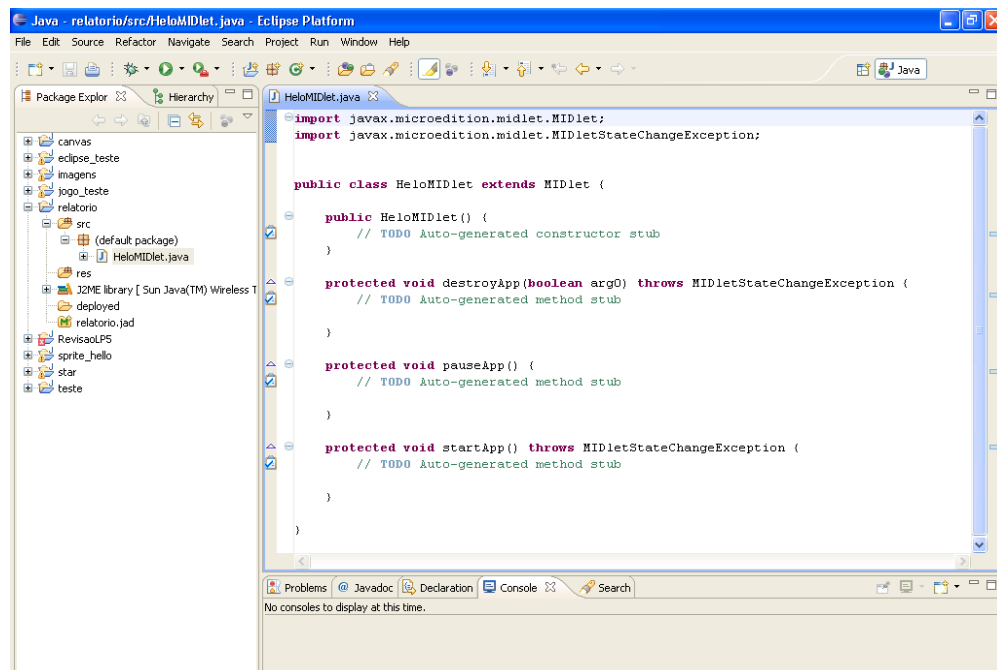
Fonte: Elaborado pelo autor, 2009

Figura 53. Tela para criação da *MIDlet*



Fonte: Elaborado pelo autor, 2009

Figura 54. Configuração da *MIDlet*



Fonte: Elaborado pelo autor, 2009

Figura 55. Nova *MIDlet* criada – com os principais *imports*, seus métodos básicos e o método construtor

Abaixo segue o código da *midlet*:

```
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Form;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

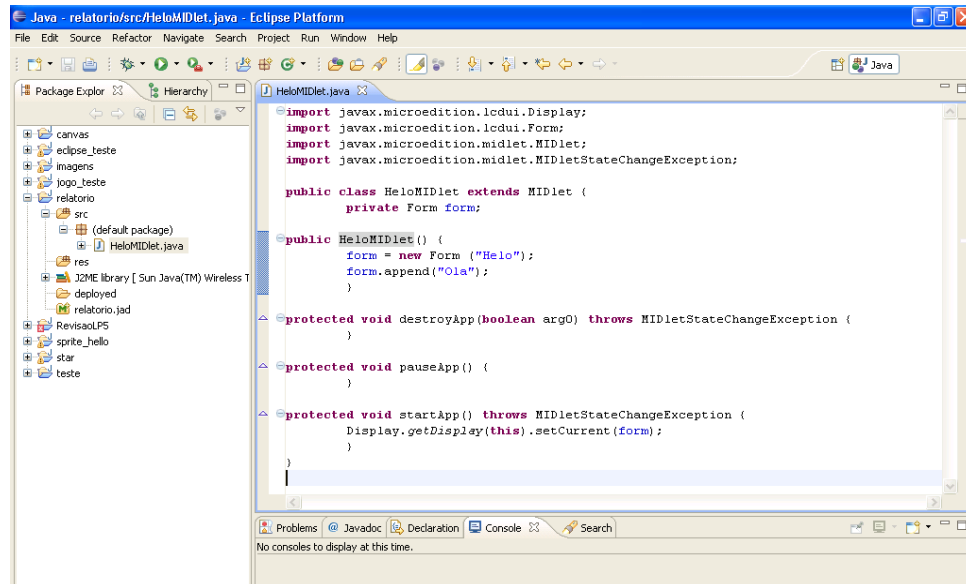
public class Helo extends MIDlet {
    private Form form;

    public Helo() {
        form = new Form ("Helo");
        form.append("Ola");
    }

    protected void destroyApp(boolean unconditional) {
    }

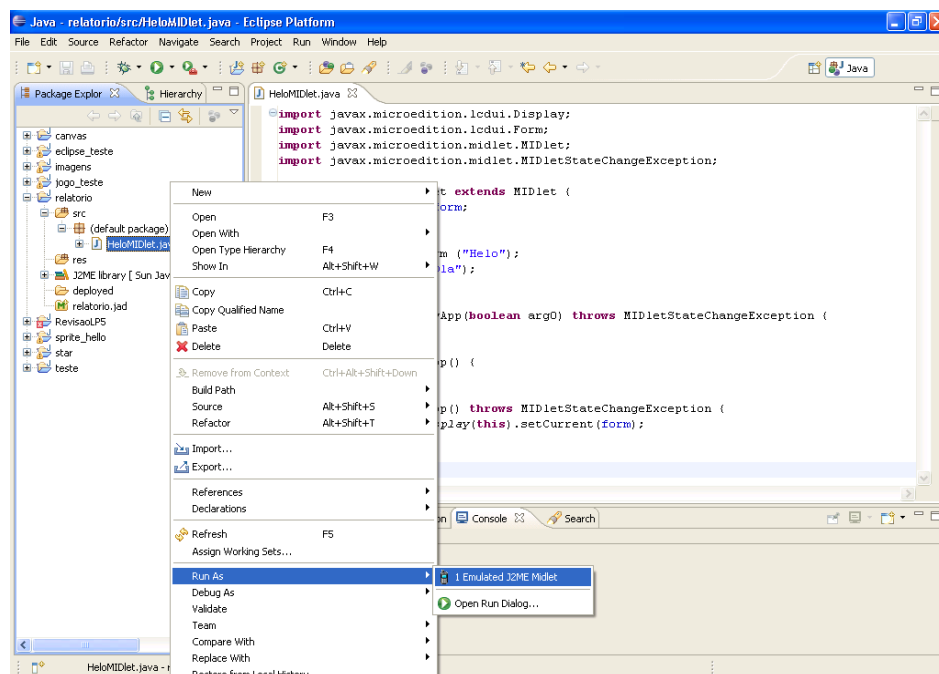
    protected void pauseApp() {
    }

    protected void startApp() throws MIDletStateChangeException {
        Display.getDisplay(this).setCurrent(form);
    }
}
```



Fonte: Elaborado pelo autor, 2009
 Figura 56. Código da *MIDlet*

Para executar a aplicação, busque no menu a opção *Run* as e, dentro desta, a opção 1 *Emulated J2ME Midlet*, o resultado deve ser algo semelhante à figura 23.



Fonte: Elaborado pelo autor, 2009
 Figura 57. Processo para execução da *MIDlet*



Fonte: Elaborado pelo autor, 2009

Figura 58. Resultado da execução da *Midlet*

ANEXO B - ENTENDENDO OS RECURSOS DA PLATAFORMA J2ME

1 RECURSOS DA PLATAFORMA J2ME

Este anexo tem por objetivo apresentar os principais recursos da plataforma J2ME utilizados na elaboração do trabalho.

O trabalho foi baseado na arquitetura J2ME com a configuração CLDC implementada com o perfil MIDP. Segundo Johnson (2008), a configuração CLDC é constituída, devido às restrições de recursos, em apenas quatro pacotes, *java.io*; *java.lang*; *java.util* e *java.microedition.io*, onde os três primeiros pacotes são versões simplificadas da versão J2SE, e somente o último é específico da J2ME. A tabela 1 apresenta uma breve descrição desses pacotes.

Quadro 2. Pacotes da configuração CLDC

Pacote	Descrição
<i>java.io</i>	Tratamento de entrada e saída de dados usando <i>streams</i> (abstração de um canal de comunicação entre o programa e a fonte de dados).
<i>java.lang</i>	Classes básicas da linguagem Java.
<i>java.util</i>	Classes de utilidades genéricas, como algumas estruturas de dados e classes para manipulação de dados.
<i>java.microedition.io</i>	Exclusivo da J2ME; inclui classes de conexões genéricas, que permitem manipular diversos tipos de conexões.

Fonte: Johnson, 2008.

Ainda segundo Johnson (2008), o perfil MIDP é composto pela API *microedition*, um conjunto de pacotes exclusivos para este perfil, formados pelos seguintes pacotes:

- a) *javax.microedition.midlet.** - define as aplicações MIDP e a interação entre a aplicação e o ambiente de execução;
- b) *javax.microedition.lcdui.** - fornece as classes para implementação da interface gráfica;
- c) *javax.microedition.lcdui.game.** - fornece as classes para o desenvolvimento de jogos;
- d) *javax.microedition.rms.** - fornece as classes usadas para implementar armazenamento de dados no dispositivo;
- e) *javax.microedition.pki.** - permite uso de autenticação para comunicações seguras.

Neste projeto foram aplicadas as API's *midlet*, *lcdui* e *lcdui.game*. A seguir são apresentadas as classes utilizadas no projeto com as suas respectivas explicações.

1.1 Pacote *javax.microedition.midlet.**

1.1.1 Classe *Midlet*

A classe *Midlet* trata-se de uma classe abstrata que pertence ao pacote *javax.microedition.midlet.**, responsável pela construção dos aplicativos e pelo gerenciamento do ciclo de vida da aplicação, através de seus métodos.

Um *MIDlet* é um aplicativo Java projetado para ser executado em um dispositivo móvel. Mais especificamente, um *MIDlet* tem como classes Java básicas a CLDC e o MIDP. (MUCHOW, 2004, p. 23).

Um *midlet* apresenta três estados: ativo, pausado e destruído, que são implementados através dos métodos básicos:

- a) método *startApp* – chamado pelo gerenciador de aplicativo para ativar a *midlet*;
- b) método *pauseApp* – é chamado pelo gerenciador de aplicativo para pausar a *midlet*;
- c) método *destroyApp* – é chamado pelo gerenciador de aplicativo para destruir a *midlet*.

No momento da criação da *midlet* os três métodos são criados, juntamente com o método construtor. Estes métodos são os principais e podem ser observados dentro do desenvolvimento do Jogo da Memória.

Abaixo segue parte de um código que exemplifica a implementação dos métodos citados.

```
package jogoMidlet;

public class Jogo extends MIDlet implements CommandListener {

    Partida partida;
    private Display display;
    private Form menu;
    private Command voltar;
    private Command sair;
    private Command jogar;

    //método Construtor
    public Jogo() {
        getDisplay().setCurrent (get_Menu());
    }

    //método chamado pelo sistema para ativar a Midlet
    protected void startApp(){
    }

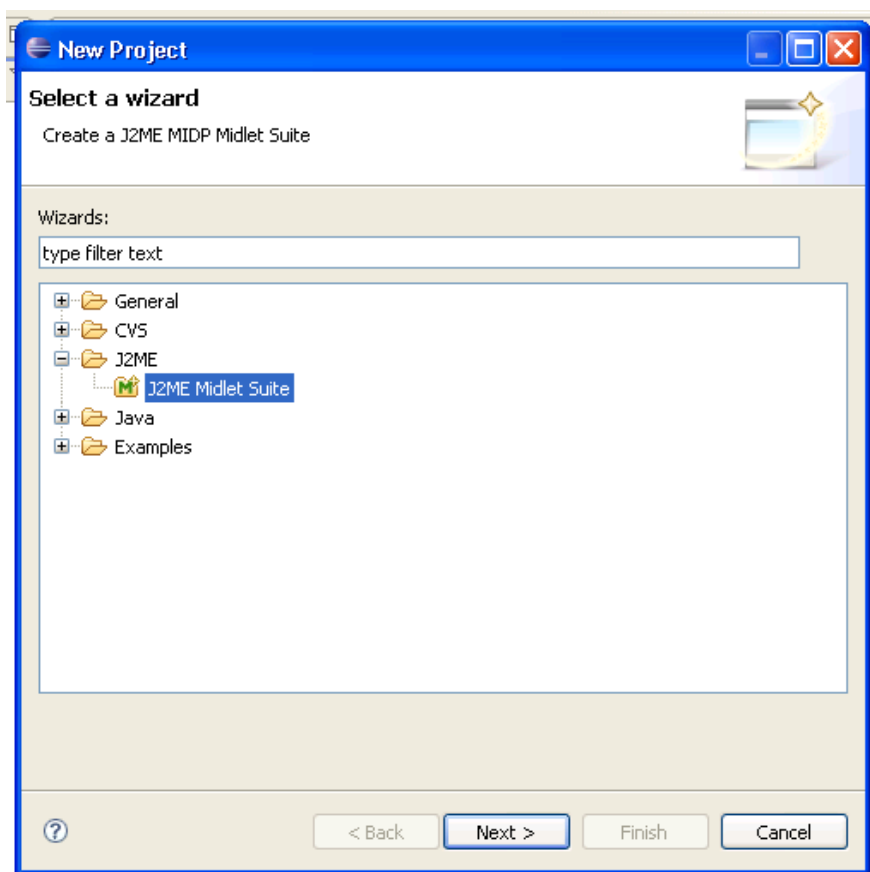
    // método chamado pelo sistema para pausar a Midlet
    protected void pauseApp() {
    }

    // método /chamado pelo sistema para destruir a Midlet
    protected void destroyApp(boolean b){
        exit();
    }
}
```

1.1.2 Midlet Suite

É o conjunto de arquivos e recursos que fazem parte do *midlet*, é o projeto onde será criado o *midlet*. É necessário criar o *Midlet Suite* antes de criar-se o *midlet*.

A Figura 1 apresenta a tela onde será criada a nova *Midlet Suite*.



Fonte: Elaborado pelo autor, 2009

Figura 59. Tela de criação do *Midlet Suite*

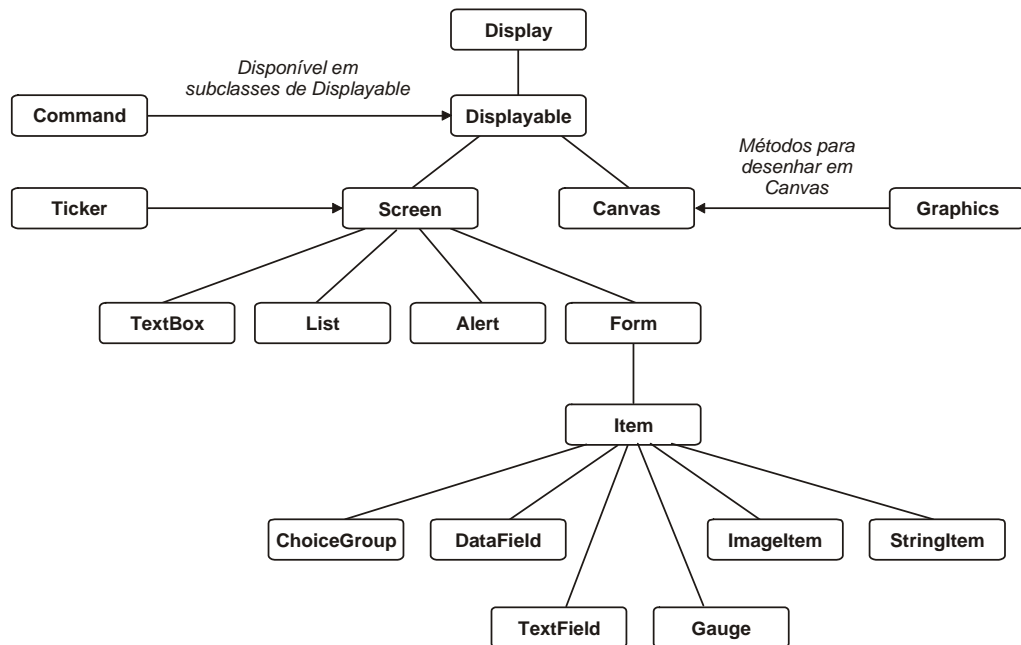
1.2 Pacote *javax.microedition.lcdui.**

Este é o pacote que contém a biblioteca responsável por definir os componentes disponíveis para serem utilizados na elaboração de uma interface para dispositivos com perfil MIDP.

Entre as diversas classes do pacote *javax.microedition.lcdui.**, foram utilizadas no projeto em questão as seguintes: *Display*; *Displayable*; *Command*; *Form*; *Canvas*; *Graphics* e *Image*.

A Figura 2 representa a hierarquia da classe *Displayable*, exibindo suas classes, subclasses e componentes disponíveis.

Observa-se que há as classes *Screen* e *Canvas* são subclasses da classe *Displayable* e que trabalham com outras subclasses para compor suas telas.



Fonte: Muchow, 2004, p. 98

Figura 60. Hierarquia da classe *Displayable*

1.2.1 Classe *Display*

O objeto do tipo *display* representa a tela de exibição. Todo dispositivo pode ter somente um, entretanto, podem ser exibidos vários elementos (*Form*, *textBox*, etc) em um *display*. Seus principais métodos são o *getDisplay*, que retorna o objeto *display* da *midlet*; e o *setCurrent*, define qual objeto do tipo *displayable* ficará visível na tela.

O código abaixo tem como resultado final a figura que está ao seu lado, que exemplifica um objeto do tipo *display*.

Para a implementação desta tela foi necessário criar um objeto *Form* (menu) que recebeu o título de “Jogo da Memória”. Através do método *addCommand*, foram adicionados ao objeto *Form* os comandos sair e jogar, estes previamente criados. Com o método *getDisplay()* e *setCurrent()* foi definido qual objeto *display* e *displayable* será exibido, neste caso o *display* da *midlet* e o *Form* *get_Menu*.

Exemplo:

```

public Jogo() {
    getDisplay().setCurrent(get_Menu());
}

public Form get_Menu(){
    if (menu==null){
        try{
            menu = new Form ("Jogo da
Memória", null);
        }catch(NullPointerException e){
            System.out.println("Erro na criação
do jogo");}

        menu.addCommand(get_Jogar());
        menu.addCommand(get_Sair());
        menu.setCommandListener(this);
    }
    return menu;
}

```



1.2.2 Classe *Displayable*

O objeto do tipo *Displayable* representa os objetos que serão mostrados no *Display*. Possui duas subclasses: *Screen* e *Canvas*.

Os principais métodos desta classe são: o *addCommand*, que é utilizado para adicionar um comando a um objeto *displayable*; e o *setCommandListener*, usado para criar o *CommandListener* (apresentado na seção 1.2.4).

Classe *Displayable* trabalha em conjunto com a classe *Display*.

1.2.3 Classe *Command*


Esta classe também faz parte da API *microedition* e permite adicionar funções relacionadas aos objetos que estão sendo exibidos. Estes comandos são definidos de forma global usando uma variável *Command*. (JOHNSON, 2008).


O objeto *Command* contém informações sobre um evento que pode ser: *Back*, *Cancel*, *Exit*, *Help*, *Item*, *OK*, *Screen* e *Stop*, cada qual com sua função.


Para criação de um objeto *Command* utiliza-se um construtor onde são informados três parâmetros:

- a) rótulo, é o texto que será associado ao objeto *Command*;
- b) tipo, determina qual o tipo de ação desejada;
- c) prioridade, representa qual a escala de prioridade deste objeto, quanto mais alto o valor menor a prioridade.

Exemplo: `ajuda = new Command("Ajuda", Command.HELP, 1);`


rótulo


tipo


prioridade

Após a definição dos comandos, é preciso referenciar a tela onde será apresentado o objeto comando, isto ocorre através do método *addCommand*.

Exemplo: `form.addCommand(ajuda)`


formulário


método


objeto Command

Segundo Muchow (2004), a especificação de um objeto do tipo *Command* não garante a execução de nenhuma ação quando o usuário inicia um evento; independente de ser especificado o objeto, é necessário escrever o código para a execução da ação.

1.2.4 Classe *CommandListener*

É a interface que permite que a aplicação saiba quando um comando foi acionado. Para que esta interface possa ser utilizada, o *Midlet* deve ser declarado da seguinte forma:

```
public class Exemplo extends MIDlet implements
CommandListener
```

O método *CommandAction()* trabalha em conjunto com o método *CommandListener()*, é chamado quando o usuário interage com o objeto *Form*, (por exemplo quando pressiona um botão) e sua declaração apresenta-se assim:

```
public void commandAction (Command c, Displayable s)
```

Dentro deste método, o primeiro argumento determina qual operação será executada, e o segundo é útil quando é trabalhado mais de um objeto *displayable*

O método *setCommandListener()* informa qual o objeto é responsável por receber os eventos gerados, por comandos, na tela atual.

Exemplo: `form.setCommandListener(this);`

O código abaixo exemplifica a criação dos objetos *Form* (*form*) e *Command* (*sair*), e a adição do comando ao *Form*.

Se o objeto *Command* “sair” foi escolhido pelo usuário, o método *commandAction* será chamado, e os métodos *destroyApp* e *notifyDestroyed* serão executados.

Exemplo:

```
//cria a variável form do tipo Form
private Form form;

//cria a variável sair do tipo Command
private Command sair;
...
//instancia o objeto Form e nomeia de "Tela"
form = new Form ("Tela");

//instancia o objeto Command
sair = new Command ("Sair", Command.EXIT, 1);

...

//adiciona o objeto Command "Sair" ao objeto Form
form.addCommand (sair);

// recebe os eventos do objeto Form
form.setCommandListener(this);

...
public void commandAction (Command c, Displayable s)
{
    if (c == sair)
    {
        destroyApp (true);
        notifyDestroyed();
    }
}
```

1.2.5 Classe *Form*

A classe *Form* é um dos diversos objetos gráficos que compõem a classe *Screen*. Os objetos das subclasses: *ChoiceGroup*, *DataField*, *TextField*, *Gauge*, *ImageItem* e *StringItem* são exibidos em um objeto *Form*.

Cada *form* possui seus comandos e suas imagens, assim são objetos independentes e devem ser iniciados com um nome de exibição.

Exemplo: `formulario = new Form ("Primeiro Form");`

Os principais métodos do objeto *Form* são: anexar (*append()*), inserir (*insert()*) e excluir (*delete()*) componentes.

Exemplos:

```
form.delete (StringItem) //deleta uma String do Form

form.insert(textfield) //insere um campo de texto no Form

form.append new ImageItem("", cima, ImageItem.LAYOUT_CENTER, null))
// anexa uma imagem ao Form
```

1.2.6 Classe *ImageItem*

A classe *ImageItem* permite que seja especificada a forma como se deseja que uma imagem apareça na tela da aplicação, centralizada, à direita ou à esquerda. (MUCHOW, 2004).

Para isto a classe *ImageItem* possui uma lista de *layouts* disponíveis, onde se destacam alguns:

- a) *layout_default* – usa o layout padrão da implementação;
- b) *layout_left* – a imagem aparece à esquerda;
- c) *layout_right* – a imagem aparece à direita;
- d) *layout_center* – a imagem aparece centralizada horizontalmente.

Exemplo:

```
Image img = Image.createImage ("/JOGO.png");
menu.append(new ImageItem ("", img, ImageItem.LAYOUT_CENTER,
null));
//uma imagem foi criada e está sendo anexada a um Form com um
layout especificado, deve aparecer centralizada na tela.
```

1.2.7 Classe *Image*

A classe *Image* representa as imagens em J2ME, e tem como principal método o *createImage*, que possibilita a criação de imagens estáticas ou dinâmicas, de acordo com os parâmetros que são informados. (RADTKE, 2009).

Exemplo:

```
Image exemplo = Image.createImage("/algumaImagem.png");
```

As imagens utilizadas na aplicação devem encontrar-se no diretório *res* da mesma. Para o caso de ocorrer uma falha durante a execução da criação da imagem é aplicada uma exceção de I/O.

Código de criação das imagens da aplicação:

```
try{
    imagem[0]=Image.createImage("/ADMd.png");
    imagem[1]=Image.createImage("/BIOd.png");
    imagem[2]=Image.createImage("/CCd.png");
    imagem[3]=Image.createImage("/EFd.png");
    imagem[4]=Image.createImage("/ENFd.png");
    imagem[5]=Image.createImage("/FISIOd.png");
    imagem[6]=Image.createImage("/PSId.png");
    imagem[7]=Image.createImage("/QUId.png");
    imagem[8]=Image.createImage("/TECd.png");
    imagem[9]=Image.createImage("/TOd.png");
    imagemFundo = Image.createImage("/VERSO.png");
}catch (Exception e){
    e.printStackTrace();
}
```

1.2.8 Classe *Graphics*

Fornece as primitivas de desenho que permitem criar linhas, arcos, retângulos entre outras formas e também incluir textos e imagens.

“Um objeto *Graphics* é o instrumento de desenho em um componente *Canvas*.” (Muchow, 2004, p. 228).

Segundo Muchow (2004), esta classe possui mais de 30 métodos, que executam operações desde desenhar arcos, caracteres, imagens, retângulos, linhas, strings até especificar fontes e cores.

No projeto do Jogo da Memória, foram utilizados alguns métodos desta classe para desenhar as imagens, a pontuação e o cursor.

1.2.9 Classe *Canvas*

Como já foi dito anteriormente a classe *Displayable* possui duas subclasses: *Screen* e *Canvas*. A primeira permite que sejam geradas telas de alto nível e seus elementos gráficos são predefinidos, já a classe *Canvas* permite criar telas onde é possível manipular eventos de baixo nível e gerar chamadas para desenho na tela, usando a classe *Graphics*. (JOHNSON, 2008).

Através da classe *Canvas*, é possível manipular eventos de teclas, como pressionar, soltar e repetir. Existem duas classificações para as teclas: teclas-padrão, são as teclas básicas de todo dispositivo, representadas pelos números de 0 a 9, # e *. As teclas de jogos são as teclas comuns usadas com mapeamento para jogos (*UP*, *DOWN*, *FIRE*, *LEFT*, *RIGHT*), estas foram as utilizadas no projeto.

Dentre os diversos métodos oferecidos pela classe, destacam-se alguns:

- a) *keyPressed* – chamado quando uma tecla é pressionada;
- b) *keyReleased* – chamado quando uma tecla é solta;
- c) *paint* – desenha a tela *Canvas*;
- d) *repaint* – solicita o redesenho da tela *Canvas*;
- e) *getGameAction* – retorna a ação de jogos associada com a tela usada.

1.3 Pacote *javax.microedition.lcdui.game*. *

Este é o pacote que possui as classes específicas para o desenvolvimento de jogos.

1.3.1 Classe *GameCanvas*

Segundo Johnson (2008), a classe *GameCanvas* trata-se de uma subclasse de *Canvas* que complementa suas funcionalidades, permitindo outra forma de manipulação de eventos, muito utilizada para criação de jogos.

O método *getGraphics*, obtém o objeto *Graphics*, que possibilita controlar o que será desenhado. O método *flushGraphics* descarrega o desenho na tela.

Outro método importante dentro desta classe é o *getKeyStates*, que obtém o estado das teclas, ou seja, identifica qual tecla está sendo pressionada. O uso das teclas pode ser representado da seguinte forma:

- a) *UP_PRESSED* – tecla *UP* pressionada;
- b) *DOWN_PRESSED* – tecla *DOWN* pressionada;
- c) *LEFT_PRESSED* – tecla *LEFT* pressionada;
- d) *RIGHT_PRESSED* – tecla *RIGHT* pressionada;
- e) *FIRE_PRESSED* – tecla *FIRE* pressionada.

A utilização de *threads*² é outro recurso útil na criação de jogos, pois permite verificar o uso das teclas, escolher próximos movimentos e chamar a renderização da tela. (JOHNSON, 2008).

1.3.2 Classe *Sprite*

Um objeto *Sprite* é um elemento animado que representa uma imagem gráfica ou uma série delas, e são criados a partir de uma imagem.

A criação de um *sprite* utiliza basicamente dois métodos construtores:

- a) `public Sprite (Image Image)` – *sprites* estáticos;
- b) `public Sprite (Image Image, int frameWidth, int frameHeight)` – para criar *sprites* animados, onde *frameWidth* representa a largura do frame e *frameHeight* representa a altura do frame.

Exemplo:

```
try{
    Image teste = Image.createImage ("/imagem.png");
    Sprite = new Sprite (image, 16, 16);
} catch (java.io.IOException e){}
```

A classe *Sprite* possui suas próprias funcionalidades de desenho, o método empregado para desenhar um objeto *sprite* é o *paint*, que é assim declarado: `public void paint (Graphics g)`.

O método que define o posicionamento do *sprite* é o *setPosition*, que segue a seguinte declaração: `public void setPosition (int X, int Y)`, onde X representa a coordenada X e Y a coordenada Y.

Assim o código que desenha um *sprite* é representando da seguinte forma:

```
exemplo.setPosition (100, 100);
exemplo.paint (g);
```

Na aplicação desenvolvida, estes métodos foram utilizados para desenhar as cartas do jogo, como se verifica no código abaixo:

² “...forma de um processo dividir a si mesmo em duas ou mais tarefas que podem ser executadas simultaneamente.” (WIKIPÉDIA, 2009).

Essa parte do código representa a criação de dois objetos *Sprite*, o primeiro (`imagens[i]`), o *array* de imagens que compõe a coleção de imagens do jogo e o segundo (`imagemVerso`), a imagem que é utilizada no verso das cartas.

```
...
    }
    for (int i =0; i<nivel.getQtdeCartas(); i++){
        imagens[i] = new Sprite
            (nivel.getImagem(i),nivel.TAMANHO_COMPRIMENTO_IMAGEM,
            nivel.TAMANHO_ALTURA_IMAGEM);
    }
    imagemVerso = new Sprite
        (nivel.getImagemFundo(),nivel.TAMANHO_COMPRIMENTO_IMAGEM,
        nivel.TAMANHO_ALTURA_IMAGEM);

//criação do Sprite
```

Esta outra parte do código apresenta o momento em que o objeto *Sprite* é desenhado, através dos métodos *paint* e *setPosition*.

Caso as cartas atendam a uma determinada condição, estas serão desenhadas com o verso para cima, se não serão desenhadas com as figuras para cima, na posição especificada pelo método *setPosition*, onde o valor de X é representado pelo comprimento da imagem mais o valor de deslocamento do eixo X e o valor de Y é representado pelo altura da imagem mais o valor de deslocamento do eixo Y.

```
public void desenhar(Graphics g){

for(int x=0; x<5; x++)
for(int y=0; y<4; y++){
    if(cartas[x][y].estaVirada()){
        imagemVerso.setPosition(
            cartas[x][y].obterX()*nivel.TAMANHO_COMPRIMENTO_IMAGEM
            +deslocamentoX,

            cartas[x][y].obterY()*nivel.TAMANHO_ALTURA_IMAGEM
            +deslocamentoY);
        imagemVerso.paint(g);
    }else{
        imagens[cartas[x][y].obterValor()].setPosition(
            cartas[x][y].obterX()*nivel.TAMANHO_COMPRIMENTO_IMAGEM
            +deslocamentoX,
            cartas[x][y].obterY()*nivel.TAMANHO_ALTURA_IMAGEM
            +deslocamentoY);
        imagens[cartas[x][y].obterValor()].paint(g);
    }
} //desenha cartas
```



Fonte: Elaborado pelo autor, 2009

Figura 61. Tela com as cartas desenhadas

ANEXO C - Como instalar jogos no celular

1 INTRODUÇÃO

Segundo Johnson (2008), para o processo de instalação de jogos em dispositivos móveis é necessário que sejam executados dois procedimentos:

- a) a geração dos arquivos .jar e .jad;
- b) a transferência dos arquivos gerados para o dispositivo.

O primeiro procedimento, de gerar os arquivos, dentro do Eclipse, ocorre da seguinte forma:

Com o botão direito do mouse, clique no nome do projeto, escolha a opção J2ME, depois *Create Package*. Isso gerará os arquivos .jar e .jad necessários para execução no dispositivo. Esses arquivos serão copiados para o diretório *deployed*, dentro do projeto. (JOHNSON, 2008, p. 324).

Entretanto alguns fabricantes não executam aplicativos gerados pelo Eclipse, é preciso recompilar o código no ambiente de desenvolvimento do fabricante.

De acordo com Muchow (2004), *Java Archive* (JAR) consiste em um arquivo onde são empacotadas as classes Java e os recursos da *midlet*, para distribuição. E o arquivo *Java Application Descripton* (JAD) trata-se de um arquivo descritor, que fornece informações sobre a *midlet*, e pode ser acessado pelo gerenciador da aplicação.

Dentro do arquivo JAR existe ainda armazenado um outro arquivo, o Manifesto, cuja função é descrever o conteúdo do JAR. Nele podem ser definidos diversos atributos, entretanto seis são obrigatórios, para que o JAR possa ser carregado pelo gerenciador de aplicativos, são eles:

- a) *MIDlet-Name* (nome do conjunto de *MIDlets*);
- b) *MIDlet-Version* (número da versão da *MIDlet*);
- c) *MIDlet-Vendor* (quem desenvolveu a *MIDlet*);
- d) *MIDlet-<n>* (uma entrada para cada *MIDlet* no arquivo JAR) - (referência a uma *MIDlet* específica dentro de um conjunto de *MIDlets*);
- e) *MicroEdition-Profile* (qual perfil do J2ME é exigido pela(s) *MIDlet(s)*);

- f) *Micro-Edition-Configuration* (qual configuração do J2ME é exigida pela(s) *MIDlet(s)*).

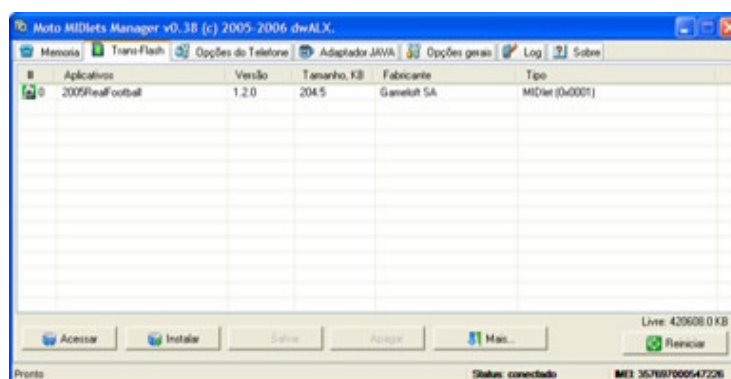
O segundo procedimento, a transferência e instalação dos arquivos no celular, pode variar de modelo para modelo; aqui serão apresentados os processos realizados em algumas marcas.

2 MODELOS

2.1 MOTOROLA

Os programas utilizados são:

- a) *Motorola Phone Tools*: permite enviar imagens, sons e vídeos;
- b) *Motorola PST*: necessário para instalar os *drivers* P2K;
- c) *P2K*, *P2KTools*, *RSDLite*: é necessário um destes aplicativos para que os *drivers* P2K reconheçam o celular;
- d) *Motomidman*: utilizado para instalar jogos e aplicativos no celular. Há outras opções, mas é recomendado o *Motomidman* pela facilidade de usar, serve para toda série P2K. Abaixo uma imagem do programa em funcionamento



Fonte: Elaborado pelo autor, 2009

Figura 62. Tela do programa Motomidman

2.2 NOKIA

Para esta marca de celular é utilizado o programa *Nokia PC Suite*, que possibilita fazer backup do celular, armazenar imagens e vídeos do

telefone no PC, enviar mensagens de texto, usar o celular como modem para conectar o computador à internet, criar e organizar a agenda de contatos, converter faixas de música para um formato que possa ser tocado no telefone, instalar jogos e outros programas do PC para o telefone, entre outras coisas.

Normalmente os celulares Nokia são acompanhados de um CD-ROM contendo o *Nokia PC Suíte*, mas também pode ser baixado.

Para instalar um aplicativo no telefone, são executados os seguintes passos:

- a) Fazer o *download* dos arquivos *.SIS, *.SISX, ou *.JAR e *.JAD do game que deseja instalar;
- b) Conectar o telefone ao computador;
- c) No Windows Explorer, clicar duas vezes no arquivo *.JAR, *.SIS ou *.SISX a ser instalado, e o aplicativo será instalado no telefone.

2.3 SONY ERICSSON

O *My Phone Explorer* é um programa gratuito, utilizado pelos celulares Sony Ericsson, que possibilita incluir/excluir nomes na agenda, fazer ligações, baixar fotos, enviar jogos e aplicativos.

Para aplicação do programa *My Phone Explorer*, é necessário realizar o processo a seguir:

- a) baixar o programa;
- b) instalar o programa e depois reiniciar o PC;
- c) Conectar o cabo de dados no celular e no PC e, em seguida, executar o *My Phone Explorer* e clicar em *connect*.

Para enviar jogos do PC para o celular, clicar com o botão direito do mouse sobre o arquivo, depois escolher a opção “enviar para o telefone”; automaticamente ele encaminhará para a pasta de jogos.

Caso não consiga instalar os jogos, filmes, toques, vídeos e outras coisas ou não possuir o cabo USB, existe a opção de receber direto no celular.

This document was created with Win2PDF available at <http://www.daneprairie.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.