

1

Introdução aos computadores, à Internet e à World Wide Web



OBJETIVOS

- Neste capítulo, você aprenderá:
- Conceitos básicos de hardware e software.
- Conceitos básicos de tecnologia de objeto, como classes, objetos, atributos, comportamentos, encapsulamento, herança e polimorfismo.
- Os diferentes tipos de linguagens de programação.
- As linguagens de programação mais amplamente utilizadas.
- Um ambiente de desenvolvimento Java típico.
- O papel do Java no desenvolvimento de aplicativos cliente/servidor distribuídos para a Internet e a Web.
- A história da linguagem-padrão da indústria de projeto orientado a objetos, a UML.
- A história da Internet e da World Wide Web.
- Como fazer *test-drive* de aplicativos Java.



1.2 O que é um computador?

- **Computador**
 - Realiza computações e toma decisões lógicas.
 - Milhões/bilhões de vezes mais rápido do que os seres humanos.
- **Programas de computador**
 - Conjuntos de instruções para as quais o computador processa dados.
- **Hardware**
 - Dispositivos físicos do sistema de computador.
- **Software**
 - Programas que executam em computadores.



1.3 Organização do computador

- **Seis unidades lógicas de sistema de computador:**
 - **Unidade de entrada**
 - **Mouse, teclado.**
 - **Unidade de saída**
 - **Impressora, monitor, alto-falantes do áudio.**
 - **Unidade de memória**
 - **Retém a entrada e as informações processadas.**
 - **Unidade de aritmética e lógica (*arithmetic and logic unit* – ALU)**
 - **Realiza cálculos.**
 - **Unidade central de processamento (*central processing unit* – CPU)**
 - **Supervisiona a operação de outros dispositivos.**
 - **Unidade de armazenamento secundária**
 - **Unidades de disco, unidades de disquete.**



1.4 Primeiros sistemas operacionais

- **Processamento em lote**
 - Um trabalho (tarefa) de cada vez.
 - Sistemas operacionais desenvolvidos.
 - Programas para tornar mais conveniente o uso dos computadores.
 - Alternação entre os trabalhos facilitada.
- **Multiprogramação**
 - Trabalhos “simultâneos”.
 - Sistemas operacionais de compartilhamento de tempo.



1.5 Computação pessoal, distribuída e cliente/servidor

- **Computação pessoal**
 - Computadores para uso pessoal.
- **Computação distribuída**
 - Computação realizada entre vários computadores.
- **Computação cliente/servidor**
 - Servidores oferecem um armazenamento comum para programas e dados.
 - Clientes acessam programas e dados provenientes do servidor.



1.6 Internet e World Wide Web

- **Internet**

- **Desenvolvida há mais de quatro décadas com financiamento do DoD (Departamento de Defesa dos Estados Unidos).**
- **Originalmente projetada para conectar alguns poucos sistemas de computador importantes.**
- **Agora acessível a centenas de milhões de computadores no mundo.**

- **World Wide Web (WWW)**

- **Permite localizar/visualizar documentos baseados em multimídia.**



1.7 Linguagens de máquina assembly e de alto nível

- **Linguagens de máquina**
 - “Linguagem natural” do computador definida pelo seu projeto de hardware.
 - Dependente de máquina.
- **Linguagens assembly**
 - Abreviações semelhantes ao idioma inglês representam operações de computador.
 - Programas tradutores assemblers convertem programas de linguagem assembly em linguagem de máquina.
- **Linguagens de alto nível**
 - Permitem escrever instruções mais semelhantes à língua inglesa.
 - Contêm operações matemáticas comumente utilizadas.
 - Os compiladores convertem a linguagem de máquina.
- **Interpretadores**
 - Executam programas de linguagem de alto nível sem compilação.



1.8 História do C e do C++

- **C++**
 - **Evoluiu a partir do C.**
 - **Evoluiu a partir do BCPL e do B.**
 - **Fornece capacidades de programação orientada a objetos.**
- **Objetos**
 - **Componentes reutilizáveis de software que modelam itens do mundo real.**



1.9 História do Java

- **Java**
 - **Originalmente desenvolvido para dispositivos eletrônicos inteligentes de consumo popular.**
 - **Depois utilizado para criar páginas da Web com conteúdo dinâmico.**
 - **Agora também utilizado para:**
 - **Desenvolver aplicativos corporativos de larga escala.**
 - **Aprimorar funcionalidades de servidores Web.**
 - **Fornecer aplicativos para dispositivos de consumo popular (telefones celulares etc.)**



1.10 Bibliotecas de classe do Java

- **Programas Java**

- Consistem em partes chamadas *classes*, as quais incluem *métodos* que realizam tarefas e retornam informações ao concluir.
- Programadores podem criar classes e métodos para construir programas Java.

- **O Java oferece bibliotecas de classe**

- Conhecidas como Java APIs (Application Programming Interfaces) ou APIs do Java.



Observação de engenharia de software 1.1

- **Utilize uma abordagem de blocos de construção para criar programas.**
- **Evite reinventar a roda — utilize partes existentes onde for possível.**
- **Chamada de *reutilização de software*, essa prática é fundamental para a programação orientada a objetos.**

Observação de engenharia de software 1.2

Ao programar em Java, você geralmente utilizará os seguintes blocos de construção:

- classes e métodos de bibliotecas de classe;
- classes e métodos que você mesmo cria; e
- classes e métodos que outros criam e se tornam disponíveis para você.

Dica de desempenho 1.1

Utilizar as classes e os métodos da API do Java – em vez de escrever suas próprias versões – pode melhorar o desempenho de programas, porque eles são cuidadosamente escritos para executar de modo eficiente. Essa técnica também diminui o tempo de desenvolvimento dos programas.



Dica de portabilidade 1.1

Utilizar as classes e os métodos da API do Java – em vez de escrever suas próprias versões – melhora a portabilidade de programa, porque esses são incluídos em cada implementação Java.

Observação de engenharia de software 1.3

As extensas bibliotecas de classe de componentes de software reutilizáveis estão disponíveis pela Internet e Web – muitas sem custo algum.



1.11 FORTRAN, COBOL, Pascal e Ada

- **FORTRAN**
 - **FORmula TRANslator**
- **COBOL**
 - **COmmon Business Oriented Language**
- **Pascal**
 - **Programação estruturada**
- **Ada**
 - **Multitarefa**



1.12 BASIC, Visual Basic, Visual C++, C# e .NET

- **BASIC**
 - **Beginner's All-Purpose Symbolic Instruction Code.**
- **.NET**
 - **Plataforma .NET.**
- **Visual Basic .NET**
 - **Baseada no BASIC original.**
- **Visual C++ .NET**
 - **Baseada no C++.**
- **C#**
 - **Nova linguagem baseada no C++ e no Java.**



1.13 Ambiente de desenvolvimento

Java típico

- **Programas Java normalmente passam por cinco fases:**
 - **Edição**
 - O programador escreve programa (e armazena o programa em disco).
 - **Compilação**
 - O compilador cria bytecodes a partir do programa.
 - **Carga**
 - O carregador de classe armazena bytecodes na memória.
 - **Verificação**
 - O verificador de bytecodes confirma que os bytecodes não violam restrições de segurança.
 - **Execução**
 - A JVM traduz bytecodes em linguagem de máquina.



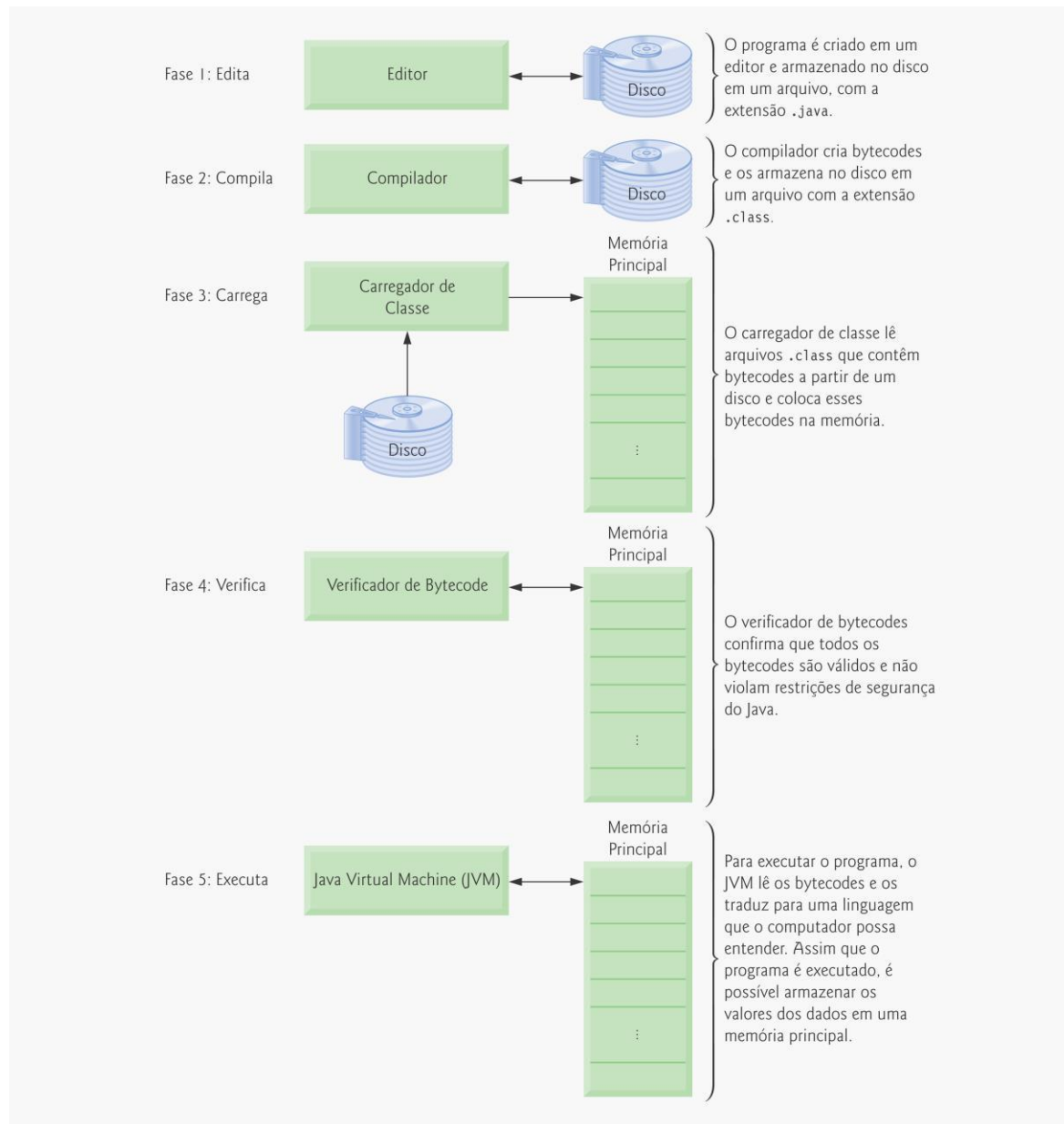


Figura 1.1 | Ambiente de desenvolvimento Java típico.



Erro comum de programação 1.1

Os erros como divisão por zero ocorrem enquanto um programa executa; portanto, eles são chamados de *erros de tempo de execução* ou *erros de runtime*.

***Erros de tempo de execução fatais* fazem com que os programas sejam imediatamente encerrados sem terem realizado seus trabalhos com sucesso.**

***Erros de tempo de execução não-fatais* permitem que os programas executem até sua conclusão, produzindo, freqüentemente, resultados incorretos.**



1.14 Notas sobre o Java e *Java: Como programar*, 6ª Edição

- **Enfatiza a clareza.**
- **Portabilidade.**



Boa prática de programação 1.1

Escreva seus programas Java de uma maneira simples e direta. Isso, às vezes, é chamado de KIS (*Keep It Simple* – mantenha a coisa simples). Não ‘estenda’ a linguagem tentando usos bizarros.



Dica de portabilidade 1.2

Embora seja mais fácil escrever programas portáveis em Java do que em outras linguagens de programação, diferenças entre compiladores, JVMs e computadores podem tornar a portabilidade difícil de alcançar.

Simplesmente escrever programas em Java não garante portabilidade.



Dica de prevenção de erro 1.1

Sempre teste os programas Java em todos os sistemas nos quais você quiser executá-los, para, assim, assegurar que eles funcionem corretamente para o público-alvo.



Boa prática de programação 1.2

Leia a documentação da versão do Java que você está utilizando. Consulte-a com frequência para certificar-se de que você está ciente da rica coleção de recursos Java e de que está utilizando esses recursos corretamente.



Boa prática de programação 1.3

Seu computador e compilador são bons professores. Caso, depois de ler cuidadosamente o manual de documentação do Java, você não esteja seguro sobre como um recurso do Java funciona, experimente-o para ver o que acontece. Estude cada erro ou mensagem de advertência/aviso que surge durante a compilação de um programa (chamados *erros de tempo de compilação* ou *erros de compilação*), e corrija os programas para eliminar essas mensagens.



Observação de engenharia de software 1.4

O J2SE Development Kit vem com o código-fonte Java. Alguns programadores gostam de ler o código-fonte das classes da API do Java para determinar como as classes funcionam e aprender técnicas de programação adicionais.



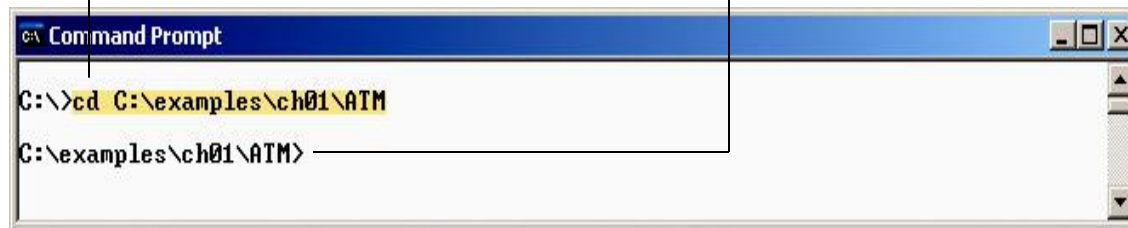
1.15 Testando um aplicativo Java

- **Test-drive de um aplicativo ATM:**
 - **Verificar a configuração do sistema.**
 - **Localizar o aplicativo ATM (Figura 1.2).**
 - **Executar o aplicativo ATM (Figura 1.3).**
 - **Inserir o número de uma conta (Figura 1.4).**
 - **Inserir um PIN (Figura 1.5).**
 - **Verificar o saldo da conta (Figura 1.6).**
 - **Retirar dinheiro da conta (Figura 1.7).**
 - **Confirmar que as informações de conta foram atualizadas (Figura 1.8).**
 - **Finalizar a transação (Figura 1.9).**
 - **Sair do aplicativo ATM.**
- **Aplicativos adicionais.**



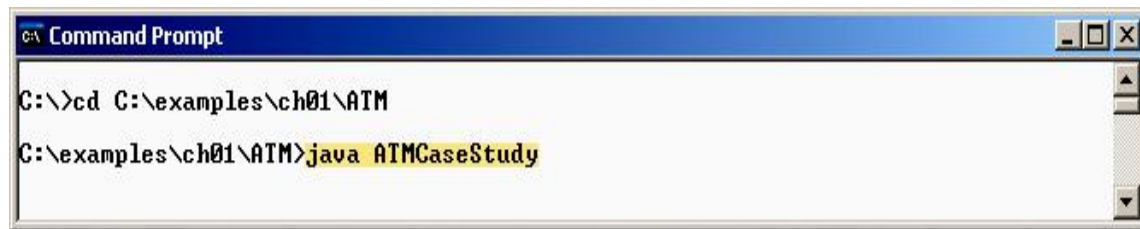
Utilizando o comando cd
para mudar de diretório.

Localização do arquivo do
aplicativo ATM.



```
CA Command Prompt
C:\>cd C:\examples\ch01\ATM
C:\examples\ch01\ATM>
```

Figura 1.2 | Abrindo um *Prompt de comando* do Windows XP e alterando diretórios.



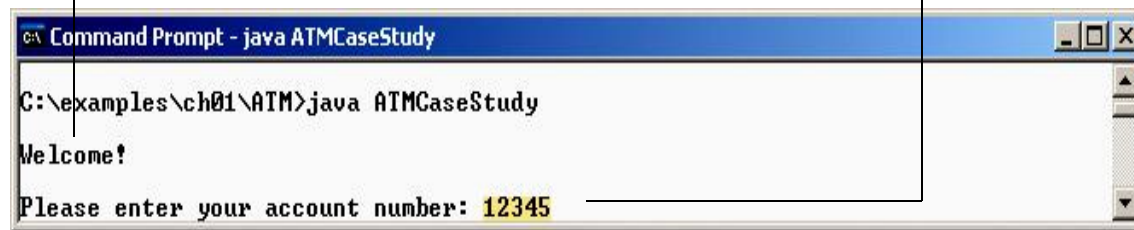
A screenshot of a Windows Command Prompt window. The title bar is blue and contains the text "C:\ Command Prompt" and standard window control buttons (minimize, maximize, close). The main area is white and contains two lines of text: "C:\>cd C:\examples\ch01\ATM" and "C:\examples\ch01\ATM>java ATMCaseStudy". The second line is highlighted in yellow. A vertical scrollbar is visible on the right side of the text area.

```
C:\>cd C:\examples\ch01\ATM
C:\examples\ch01\ATM>java ATMCaseStudy
```

Figura 1.3 | Utilizando o comando java para executar o aplicativo ATM.

Mensagem de boas-vindas do ATM.

Insira o número da conta.

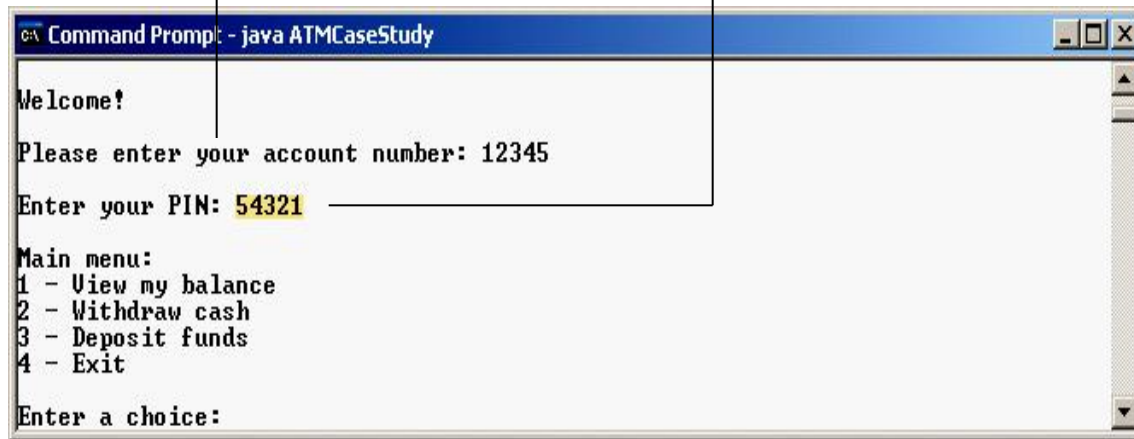


```
CA Command Prompt - java ATMCaseStudy
C:\examples\ch01\ATM>java ATMCaseStudy
Welcome!
Please enter your account number: 12345
```

Figura 1.4 | Solicitando o número de uma conta ao usuário.

Insira um PIN válido.

Menu principal do ATM.

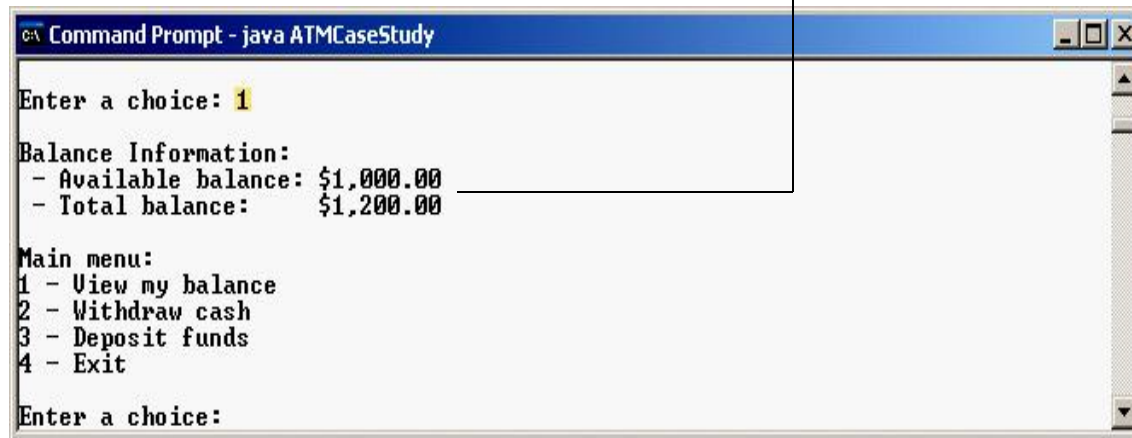


```
Command Prompt - java ATMCaseStudy

Welcome!
Please enter your account number: 12345
Enter your PIN: 54321
Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit
Enter a choice:
```

Figura 1.5 | Inserindo um número de PIN válido e exibindo o menu principal do aplicativo ATM.

Informações sobre o saldo da conta.



```
Command Prompt - java ATMCaseStudy

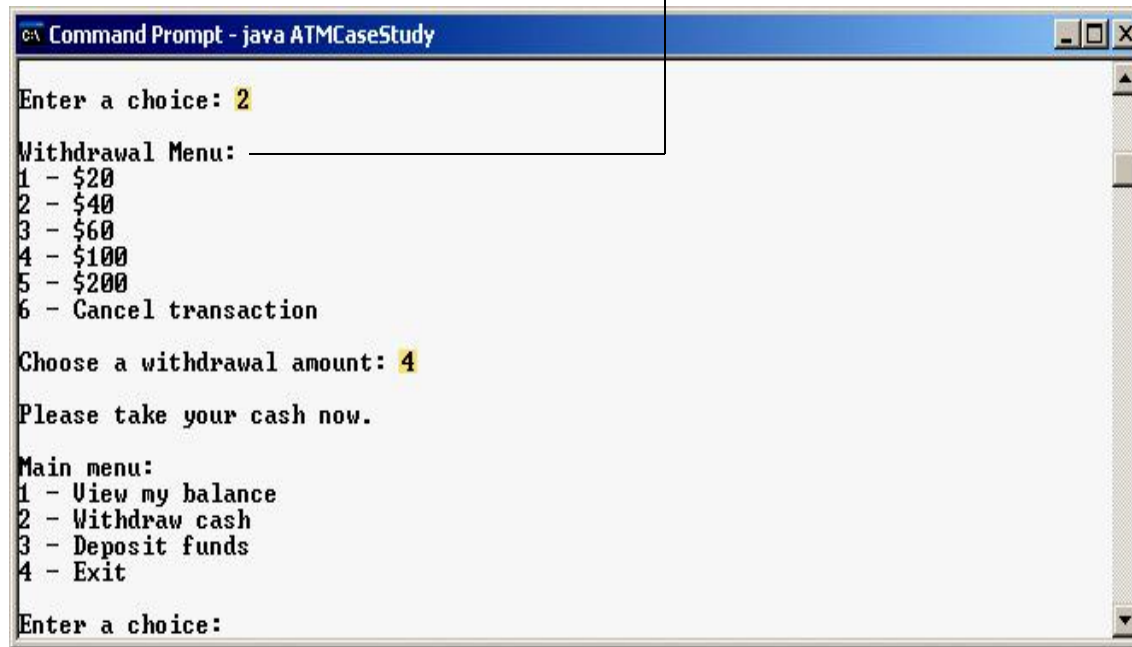
Enter a choice: 1
Balance Information:
- Available balance: $1,000.00
- Total balance:    $1,200.00

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice:
```

Figura 1.6 | O aplicativo ATM exibindo as informações de saldo da conta do usuário.

Menu de retirada do ATM.



```
Command Prompt - java ATMCaseStudy

Enter a choice: 2

Withdrawal Menu:
1 - $20
2 - $40
3 - $60
4 - $100
5 - $200
6 - Cancel transaction

Choose a withdrawal amount: 4

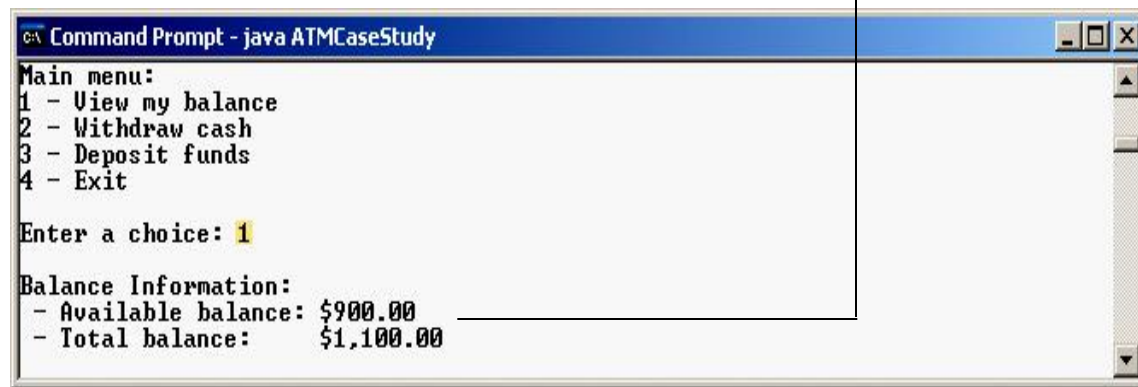
Please take your cash now.

Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

Enter a choice:
```

Figura 1.7 | Retirando dinheiro da conta e retornando ao menu principal.

Confirmando as informações atualizadas de saldo da conta após a transação de retirada.



```
C:\ Command Prompt - java ATMCaseStudy
Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit

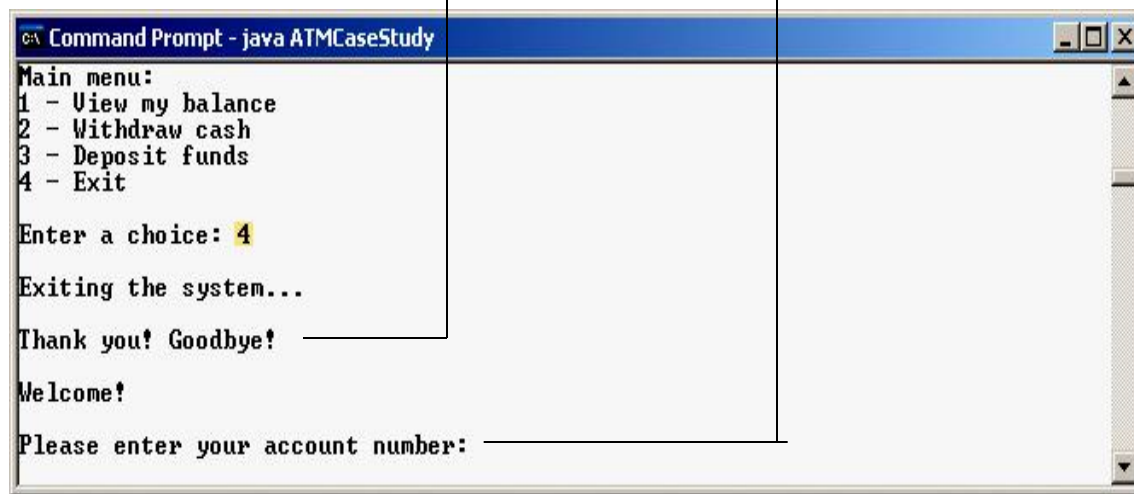
Enter a choice: 1

Balance Information:
- Available balance: $900.00
- Total balance:    $1,100.00
```

Figura 1.8 | Verificando novo saldo.

Prompt do número da conta para o próximo usuário.

Mensagem de despedida do ATM.



```
C:\ Command Prompt - java ATMCaseStudy
Main menu:
1 - View my balance
2 - Withdraw cash
3 - Deposit funds
4 - Exit
Enter a choice: 4
Exiting the system...
Thank you! Goodbye!
Welcome!
Please enter your account number:
```

Figura 1.9 | Concluindo uma sessão de transação de um ATM.

| Nome do aplicativo | Localização do capítulo | Comandos a executar |
|-----------------------------|-------------------------|---|
| Tic-Tac-Toe (jogo da velha) | Capítulos 8 e 24 | cd C:\examples\ch01\Tic-Tac-Toe java TicTacToeTest |
| Jogo de adivinhação | Capítulo 11 | cd C:\examples\ch01\GuessGame java GuessGame |
| Animador de logotipo | Capítulo 21 | cd C:\examples\ch01\LogoAnimator java LogoAnimator |
| Bola que rebate | Capítulo 23 | cd C:\examples\ch01\BouncingBall java BouncingBall |

Figura 1.10 | Exemplos de aplicativos Java adicionais encontrados em *Java: Como programar*, 6ª edição.

1.16 Estudo de caso de engenharia de software: Introdução à tecnologia de objetos e UML (obrigatório)

- **Orientação a objeto.**
- **Unified Modeling Language™ (UML™):**
 - **Linguagem gráfica que utiliza notação-padrão.**
 - **Permite a desenvolvedores representar projetos orientados a objetos.**



1.16 Estudo de caso de engenharia de software (*Continuação*)

- **Objetos**

- **Componentes reutilizáveis de software que modelam itens do mundo real.**
- **Olhe à sua volta:**
 - **Pessoas, animais, plantas, carros etc.**
- **Atributos**
 - **Tamanho, forma, cor, peso etc.**
- **Comportamentos**
 - **Bebês choram, engatinham, dormem etc.**



1.16 Estudo de caso de engenharia de software (*Continuação*)

- **Projeto orientado a objetos (*object-oriented design* – OOD)**
 - Modela objetos do mundo real.
 - Modela a comunicação entre objetos.
 - *Encapsula* atributos e operações (comportamentos):
 - Ocultamento de informações.
 - Comunicação por meio de interfaces bem definidas.
- **Linguagem orientada a objetos**
 - A programação em linguagens orientadas a objetos é chamada de *programação orientada a objetos (OOP)*.
 - Java.



1.16 Estudo de caso de engenharia de software (*Continuação*)

- **Análise e design orientados a objetos (Object-Oriented Analysis and Design – OOA/D)**
 - Essencial para programas grandes.
 - Analisa requisitos de programa e, então, desenvolve a solução.
 - UML
 - Unified Modeling Language



1.16 Estudo de caso de engenharia de software (*Continuação*)

- **História da UML**

- Criou-se a necessidade de se ter um processo com o qual abordar a OOA/D.
- Originalmente concebida por Booch, Rumbaugh e Jacobson.
- Supervisionada pelo Object Management Group (OMG).
- A versão 1.5 é a versão atual.
 - A versão 2 está em desenvolvimento.



1.16 Estudo de caso de engenharia de software (*Continuação*)

- **UML**
 - **Esquema de representação gráfica.**
 - **Permite aos desenvolvedores modelar sistemas orientados a objetos.**
 - **Flexível e extensível.**

