1- Clone o projeto em sua máquina e descreva os erros que você encontrou. Obs.: Os erros podem ser desde código, estrutura de dados, boas práticas, experiência do usuário e regras de negócio.

REGRAS DE NEGÓCIO:

1. O saldo da empresa está sendo mostrado sem o desconto da taxa;
2. O estoque de produtos não está sendo alterado.

CÓDIGO:

1. Atributos e variáveis com acentuação.

BOAS PRÁTICAS:

1. Não existe uma estrutura em pacotes para que se identifique a responsabilidade de cada classe;
2. Na classe Main há muito código. Além disso, não existe uma divisão em métodos mais específicos e de responsabilidade bem definida que facilitem a legibilidade e manutenção;
3. Não existe organização para que não se misturem códigos de regras de negócio, banco de dados e visualização de informações;
4. O método 'criarVenda' na classe Main poderia estar em uma classe relacionada apenas às vendas, e não na classe de execução da aplicação;
5. Existência de códigos boilerplate (como os getters e setters).

EXPERIÊNCIA DO USUÁRIO:

1. Menus com opções fora de ordem e/ou sem uma padronização;
2. Erros de digitação e acentuação na exibição da aplicação no terminal;
3. Exibição de stack trace ao usuário quando um erro acontece;
4. Interrupção inesperada da aplicação quando algo sai do fluxo previsto;
5. Solicitação de usuário e senha após cada operação, mesmo quando o logout não foi solicitado;
6. Não existe no menu uma opção para encerrar a aplicação;
7. Não é apresentado o preço dos produtos ao usuário no momento da escolha de produtos durante uma compra;
8. Algumas informações apresentadas na aplicação estão mal formatadas (falta de espaçamento entre informações, pontuação, sinalização da moeda (R$), formatação do valor monetário com 2 casas decimais).

2- Descreva como se estivesse repassando os ajustes para um programador.

REGRAS DE NEGÓCIO:

1. É necessário corrigir a lógica do método criarVenda para que em empresa.setSaldo() o saldo seja acrescido do valor líquido da venda, e não do valor total;
2. A lógica do método criarVenda precisa pegar a quantidade vendida de cada item do carrinho e subtrair essa quantidade do produto através de um setQuantidade de Produto (onde a lógica será: quantidade -= quantidadeVendida).

CÓDIGO:

1. Os acentos dos atributos e variáveis (como 'código') podem ser alterados para 'codigo'.

BOAS PRÁTICAS:

1. As classes poderiam ser organizadas em pacotes de acordo suas respectivas responsabilidades. Por exemplo: as classes Cliente, Empresa, Produto, Usuário e Venda estão relacionadas ao domínio do negócio, sendo assim, poderiam estar agrupadas no pacote "src/domain";
2. Poderia haver uma separação de trechos de código em métodos específicos e de responsabilidade bem definida. Após a organização em métodos, seria possível ainda criar classes específicas para agrupar esses métodos de acordo com suas responsabilidades. Por exemplo: pode haver o método 'realizarLogin', o método 'mostrarMenuCliente', 'mostrarMenuEmpresa', etc.);
3. Um pacote chamado 'service' poderia ser criado para agrupar classes que tratem da lógica de vendas, lógica de compras, etc.;
4. Seguindo a lógica dos itens anteriores, o método 'criarVenda' poderia estar agrupado em uma classe com outras lógicas e regras relacionadas às vendas (exemplo: src/ /service/VendasService);
5. Poderia ser adicionado ao projeto um gerenciador de dependências (como o Maven), e então adicionar o Lombok ao projeto para que esses códigos fossem substituídos por anotações.

EXPERIÊNCIA DO USUÁRIO:

1. Pode-se inverter o ID dos construtores empresa e empresa2 entre si para que apareçam na ordem 1 – 2 – 3;
2. Os construtores de produtos podem ser revisados, assim como as mensagens do menu;
3. Poderiam ser realizados lançamentos de exceções e tratamento de erros para que o usuário não tenha conhecimento das mensagens de erro da linguagem de programação;
4. Assim como a correção do item anterior, lançamentos de exceções e tratamento de erros possibilitariam um fluxo contínuo da aplicação

5. O método ‘executar’ faz chamadas a si mesmo após todas as operações realizadas. Como o processo de login está incluso no método ‘executar’, isso significa que o login será solicitado a cada operação realizada. Seria ideal que o login não fosse realizado dentro do método ‘executar’. Além disso, poderia ser criada uma variável do tipo boolean que se tornasse true quando um login fosse realizado, e então as operações poderiam ser executadas apenas verificando se essa variável é true, ao invés de solicitar todo o processo de login novamente;
6. Pode-se adicionar uma quarta opção no primeiro menu exibido após o login, onde a escolha dessa opção na estrutura switch executaria apenas um comando break;
7. Pode-se adicionar um x.getPreco() ao final da linha que é responsável pela exibição do código e do nome do item;
8. Pode-se criar uma variável do tipo DecimalFormat para que seja possível formatar esses valores de acordo com o desejado.

3- Em caso de erros na regra de negócio, faça um relato para a empresa que solicitou o sistema, neste relato deve ser informado o erro e porquê acontece o erro.

Não encontrei erros nas regras de negócio solicitadas, mas sim na aplicação das mesmas no código.

4- Faça o máximo de ajustes no código, de forma que as falhas sejam corrigidas. Siga a seguinte ordem para ajuste: Regra de negócio, código, boas práticas, estrutura de dados e experiência do usuário.

As correções realizadas estão descritas nos commits feitos no meu repositório.

5- Suba os ajustes no seu github (caso tenha feito apenas os descritivos, por favor desconsiderar).