# Encript

O arquivo Encript.java implementa uma classe chamada Encript que realiza a criptografia de strings usando um método de deslocamento de caracteres, também conhecido como cifra de César.

A classe faz parte do pacote client e importa java.io.Serializable para permitir a serialização, e compute.Task para definir uma tarefa que pode ser enviada para execução no servidor.

Recebe uma stringOriginal e um deslocamento, que é o número de posições que cada letra será deslocada no alfabeto.

Método encriptString: Realiza a criptografia da string. Para cada caractere: Se for uma letra, ela é deslocada pelo número especificado, respeitando a distinção entre maiúsculas e minúsculas. Caracteres que não são letras são mantidos como estão.

# IMC

O arquivo IMC.java implementa uma classe chamada IMC que calcula o Índice de Massa Corporal (IMC) de uma pessoa.

Assim como o arquivo anterior, IMC faz parte do pacote client e importa java.io.Serializable e compute.Task.

Recebe altura e peso como parâmetros para serem usados no cálculo do IMC.

# QRCodeGenerator

O arquivo QRCodeGenerator.java implementa uma classe para gerar códigos QR.

A classe está no pacote client e utiliza várias importações, incluindo classes de bibliotecas de terceiros como zxing para a geração de QR Codes.

Aceita parâmetros texto, largura, altura, e caminho que definem o conteúdo do QR Code, suas dimensões e o local de salvamento.

Método gerarQRCode: Usa a biblioteca zxing para criar um QR Code com as especificações fornecidas. O método tenta:

* Configurar o QR Code com ajustes específicos (como o conjunto de caracteres UTF-8).
* Gerar a matriz de bits do QR Code.
* Salvar o QR Code gerado em um arquivo no caminho especificado.
* Retorna o caminho absoluto do arquivo gerado ou null se ocorrer um erro.

# Senha

O arquivo Senha.java contém uma implementação para gerar senhas seguras.

A classe está no pacote client e utiliza java.io.Serializable para serialização, além de outras importações úteis para geração de senhas seguras.

Implementa Task<String>, significando que retorna uma String (a senha gerada) quando executada. A classe também é serializável.

Aceita um comprimento para a senha, verificando se ele é positivo e razoável (por exemplo, maior ou igual a 8).

Executa a geração da senha chamando o método gerarSenha.

Utiliza SecureRandom para gerar uma senha que:

* Inclui pelo menos um caractere de cada categoria (maiúsculas, minúsculas, dígitos e caracteres especiais).
* Preenche o restante até alcançar o comprimento desejado com caracteres aleatórios de todas as categorias.
* Embaralha os caracteres para garantir uma distribuição aleatória.

# Compute

O arquivo Compute.java define uma interface chamada Compute no pacote compute, que é usada para estabelecer um contrato para execução de tarefas no ambiente distribuído utilizando Java RMI (Remote Method Invocation).

Tem um parâmetro genérico Task<T> e retorna um objeto do tipo T. Isso permite que qualquer tipo de tarefa que implemente a interface Task seja passada para esta interface e executada remotamente.

# Task

O arquivo Task.java define uma interface genérica Task<T> dentro do pacote compute. Esta interface é fundamental para a estrutura das tarefas que são enviadas e executadas no ambiente distribuído usando Java RMI.

* Interface Task: É uma interface genérica, onde T representa o tipo de dado que a tarefa retorna ao ser executada.
* Método execute: Este método não possui parâmetros e retorna um objeto do tipo T. Ele define o contrato para a execução da tarefa, onde cada tarefa implementará sua lógica específica nesse método.

# Sockets

# Client

O arquivo TCPClient.java contém a implementação de um cliente TCP que interage com um servidor para executar diferentes tarefas remotamente.

A classe está no pacote sockets e usa várias classes Java para gerenciamento de redes e I/O, como Socket, ObjectInputStream, e ObjectOutputStream.

O cliente estabelece uma conexão com o servidor usando um Socket especificando o endereço e a porta (por padrão, localhost e 7896).

O cliente usa ObjectOutputStream para enviar dados ao servidor. Os dados enviados dependem dos argumentos passados ao iniciar o programa (por exemplo, a solicitação pode ser para encriptar uma string, gerar uma senha, calcular o IMC ou gerar um QR Code).

ObjectInputStream é usado para receber respostas do servidor, como strings encriptadas, valores de IMC, caminhos de arquivos de QR Codes, etc.

# Server

O arquivo TCPServer.java implementa um servidor TCP que recebe solicitações de um cliente, executa tarefas e envia os resultados de volta.

Está no pacote sockets e importa classes necessárias para a rede, I/O, e interação com objetos RMI.

Inicializa e roda um ServerSocket em uma porta específica (7896). Aceita conexões de clientes e inicia uma nova thread (Connection) para cada cliente conectado.

Classe Connection (Thread): Lida com a comunicação entre o servidor e um cliente específico.

* Leitura e escrita de objetos: Usa ObjectInputStream e ObjectOutputStream para ler solicitações do cliente e enviar respostas.
* Execução de Tarefas: Depende do tipo de solicitação (por exemplo, encript, senha, imc, qrcode), cria e executa a tarefa correspondente utilizando os objetos recebidos.
* RMI Integration: Tenta conectar-se a um serviço RMI chamado Compute para possivelmente delegar algumas tarefas para execução remota, embora detalhes específicos sobre o uso de RMI para execução das tarefas não estejam completamente visíveis no trecho de código fornecido.
* Tratamento de Exceções: Lida com várias exceções que podem surgir durante a comunicação de rede ou execução de tarefas.

# Scripts .bat

O arquivo setup.bat é um script de lote que automatiza o processo de compilação dos componentes Java da aplicação.

* Compila as interfaces Compute.java e Task.java do pacote compute.
* Verifica se houve erro na compilação (if %errorlevel% neq 0 pause), e pausa se houve algum para que o usuário possa verificar o problema.
* Empacota as classes compiladas em um arquivo JAR chamado compute.jar na pasta classes.
* Compila ComputeEngine.java do pacote engine, usando o arquivo JAR do pacote compute como parte do classpath.
* Verifica e pausa em caso de erro.
* Compila as classes Encript.java, Senha.java, IMC.java e QRCodeGenerator.java do pacote client, também usando o compute.jar e bibliotecas adicionais do ZXing (libs/core-3.5.3.jar, libs/javase-3.5.3.jar) para QRCodeGenerator.java.
* Novamente, verifica e pausa após cada compilação se houver erros.
* Compila TCPServer.java e TCPClient.java do pacote sockets, que são responsáveis pela comunicação de rede. As dependências incluem compute.jar e as bibliotecas ZXing.
* Verifica e pausa após cada compilação se houver erros.

O arquivo runTCP.bat é um script em batch que automatiza a inicialização de componentes necessários para executar o servidor TCP e RMI no ambiente Windows.