Aplicações Móveis

Lab-3

Coordenação de Engenharia Informática

Departamento de Engenharias e Tecnologias

Instituto Superior Politécnico de Tecnologias e Ciências

Nome : Marcelo Rocha - 20210032

Respostas Lab 3

b. Inspecione as threads de sua aplicação usando o depurador. Execute a aplicação no emulador, pressione o botão "Start" e abra o Android Device Monitor (ADM). No painel "Devices" do ADM (no lado esquerdo), selecione o ID da aplicação e clique no ícone "Update threads" na parte superior do painel. Identifique a thread principal e a worker thread em execução. Em seguida, pressione o botão "Stop". O que aconteceu com as threads?

R: Ao iniciar a aplicação e pressionar o botão "Start", uma worker thread é criada para executar uma contagem em segundo plano, enquanto a thread principal (UI thread) continua a gerenciar a interface do usuário. No Android Device Monitor (ADM), você verá a thread principal ativa e a worker thread em execução. Quando o botão "Stop" é pressionado, a worker thread é interrompida usando thread.interrupt(), entrando em estado TERMINATED, enquanto a thread principal permanece ativa para permitir a interação do usuário.

a. Importe o projecto SimpleImageDownload.zip para o Android Studio. Compile-o, teste-o no emulador e estude seu código. O que esta aplicação faz?

R: A aplicação Android implementa o download de uma imagem a partir de uma URL utilizando threads. Ela possui dois botões: btnDownloadFile, que inicia o download em uma nova thread, e btnDownloadFileAsync, que ainda não tem funcionalidade implementada. Ao pressionar o botão de download, a aplicação cria uma thread que executa o método

downloadImage(), que tenta baixar uma imagem da URL https://android.com/images/froyo.png. O status do download é exibido em uma TextView, e se o download for bem-sucedido, um log de sucesso é registrado. A aplicação evita travamentos na interface do usuário ao realizar o download em uma thread separada.

Saiba mais sobre filas de mensagens e comunicação entre threads estudando uma implementação simples do problema clássico de produtores-consumidores. Importe o projeto ProducerConsumerWithLooper.zip para o Android Studio. Compile-o, execute-o e entenda seu código. Interprete a saída do programa no console LogCat. No código-fonte, qual é o papel da variável do handler? O que é um Looper?

R: No código, o Handler é responsável por permitir a comunicação entre a thread Producer, que gera números aleatórios, e a thread Consumer, que processa esses números

A variável handler está associada à Thread Consumer. No método run() da Consumer, a handler é inicializada para que ela possa receber mensagens de outras threads, como a Producer.

A Producer usa o Handler para enviar mensagens contendo os números para a Consumer, que os processa verificando se são divisíveis por 2. Para que a Consumer permaneça ativa e processando essas mensagens continuamente, o Looper é utilizado. O Looper mantém a thread Consumer em execução contínua, esperando e processando mensagens na fila até que seja interrompido, ao invés de permitir que a thread termine imediatamente após a execução do seu código inicial. Isso garante que a Consumer fique disponível para responder às mensagens enviadas pelas threads Producer indefinidamente.

Inicie a aplicação e pressione o botão "Use Internal Storage". Escreva algum texto evclique em "Write". Em seguida, toque no botão "Read". Para garantir que o texto seja armazenado persistentemente, encerre e reinicie a aplicação. Em seguida, na actividade principal, selecione a mesma opção anterior e toque no botão "Read". Deve ver o seu texto. Explique como a aplicação está a executar as operações de leitura e escrita de ficheiros, estudando o código-fonte da aplicação. Qual é o nome do ficheiro de dados acessado pela aplicação?

R: Na aplicação apresentada, as operações de leitura e escrita de ficheiros são realizadas utilizando o armazenamento interno do Android. O nome do ficheiro acessado pela aplicação é "test.txt". A aplicação realiza operações de leitura e escrita no arquivo **"test.txt"** utilizando o armazenamento interno do Android. Para a escrita, o método **write()** abre o arquivo em modo privado com **openFileOutput()**, garantindo que seja acessível apenas pela aplicação. O texto inserido pelo usuário no campo de entrada é convertido em bytes e salvo no arquivo. Após a gravação, o conteúdo dos campos de entrada e saída é limpo e uma mensagem de confirmação é exibida. Já na leitura, o método **read()** utiliza **openFileInput()** para abrir o arquivo e um **Scanner** para ler seu conteúdo linha por linha, armazenando-o em um **StringBuilder**. O texto lido é exibido no campo de saída após a leitura. Em ambas as operações, os fluxos são fechados no bloco finally para garantir que os recursos sejam liberados corretamente, evitando vazamentos de memória.

Link do Github - https://github.com/marcelo365/Aplicacoes-Moveis