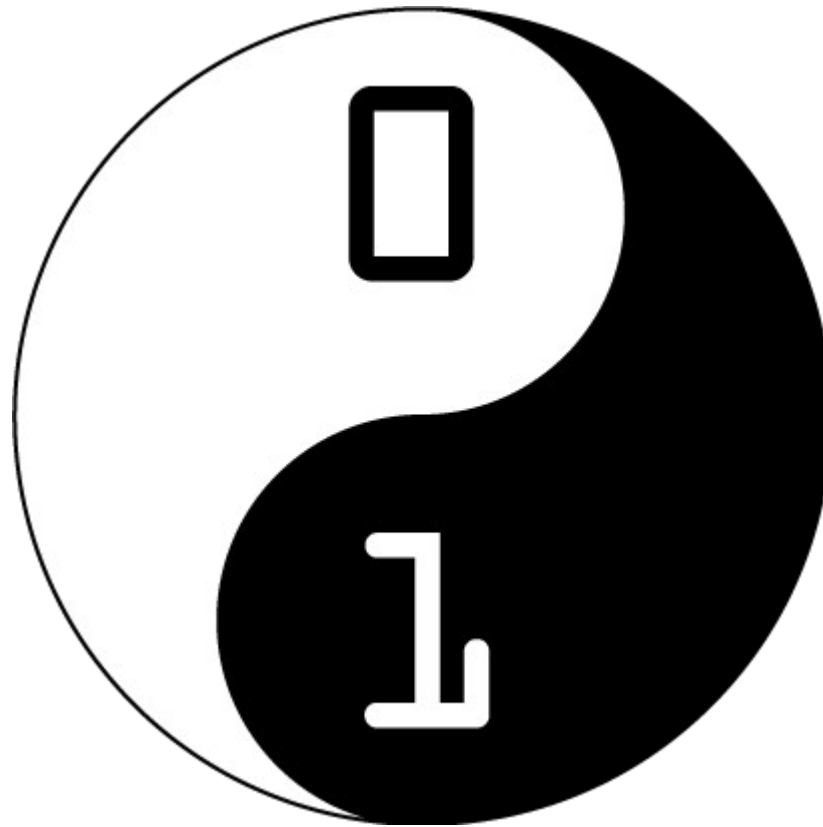
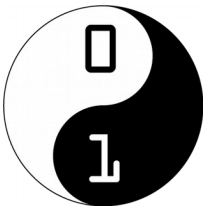


Coding Dojo



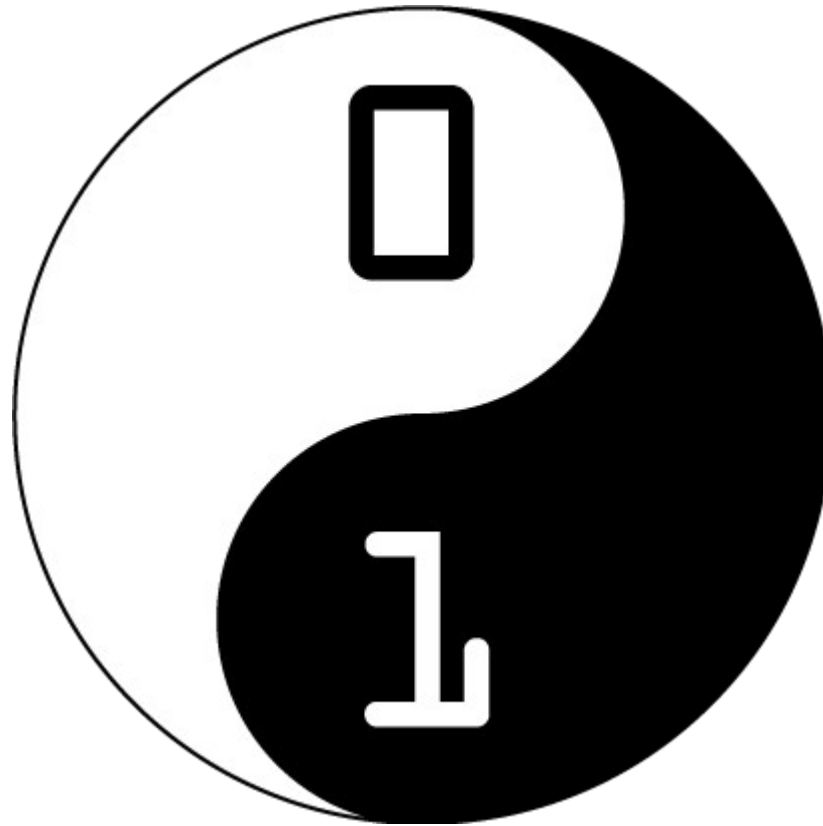
VAMOS COLETAR?



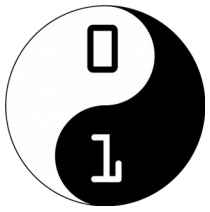
Coding Dojo - Regras

- Ponto de participação, poderá perdido se:
 - Indisciplina
 - Aluno atrasado (ver política de atraso na especificação)
 - Recusar participação como piloto/copiloto
 - Ou não querer sair do computador, quando solicitado :-)
 - Demorar para sair quando solicitado
 - Parar de participar por:
 - Uso de outro computador
 - Uso de celular
 - Usar a internet
 - Conversa

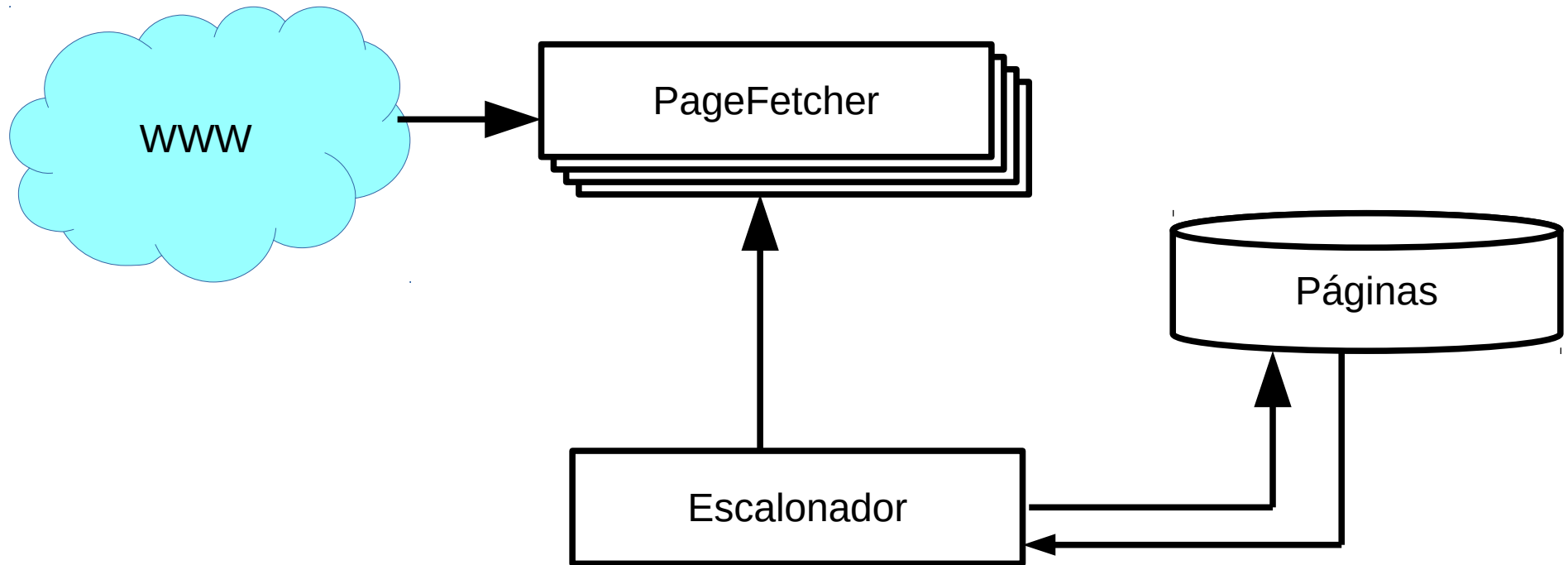
Coding Dojo

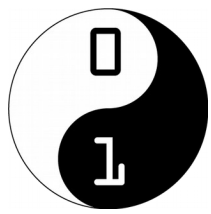


Escalonador Simples

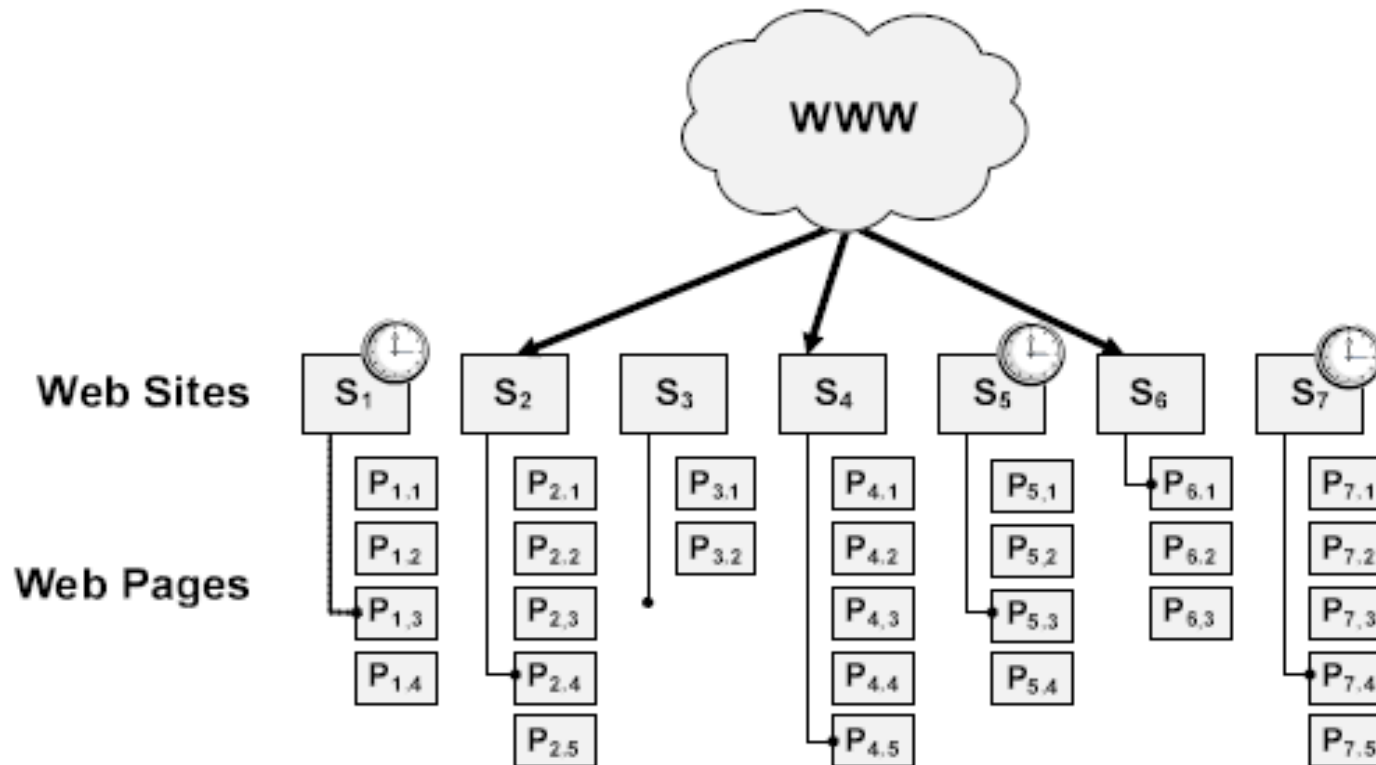


Coding Dojo - Escalonador

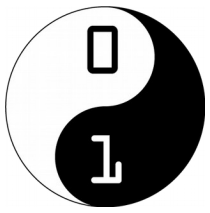




Coding Dojo – Escalonador



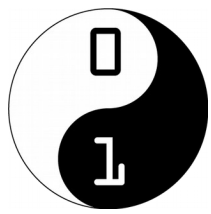
- Devemos, inicialmente:
 - Usar a classe servidor para armazenar:
 - O nome do servidor
 - Para simplificar, usaremos o hostname.
 - O horário do último acesso



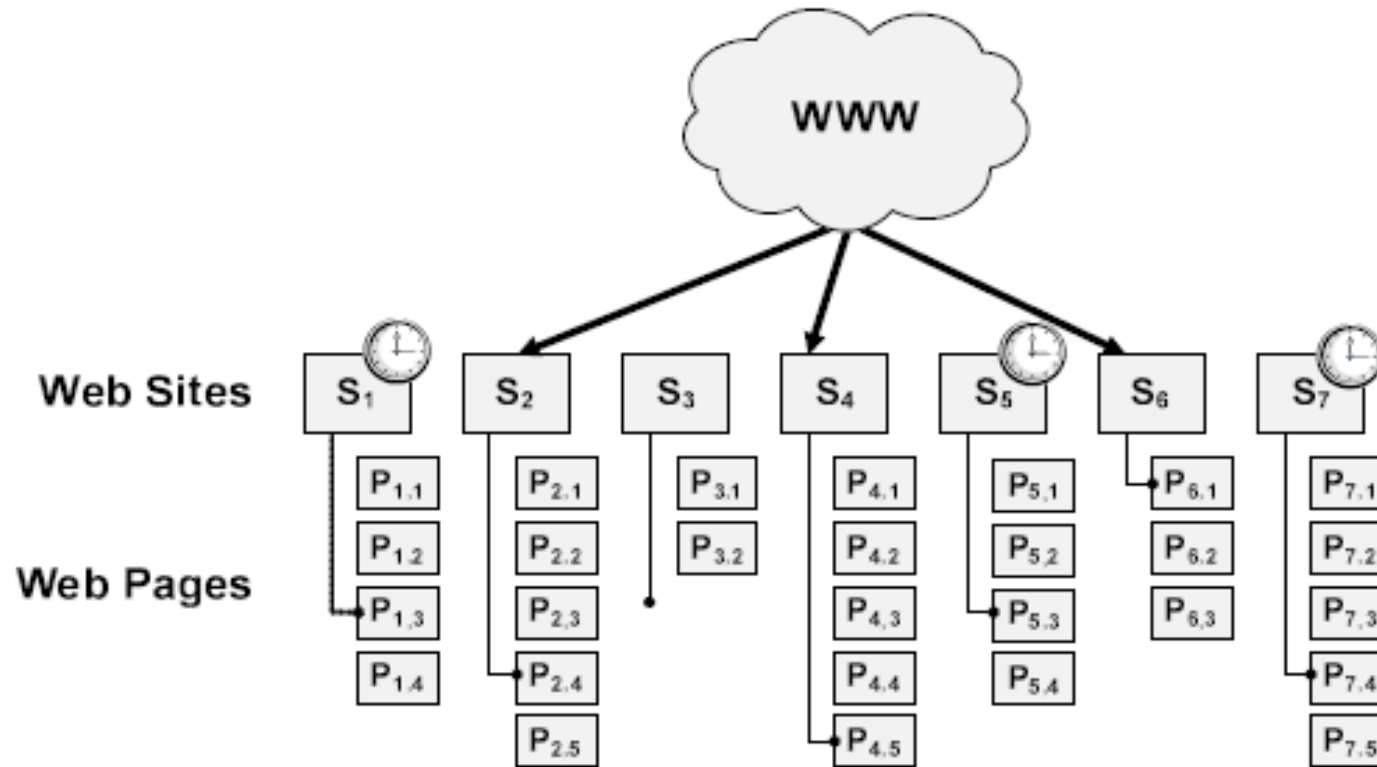
Coding Dojo – Escalonador

Classe Servidor

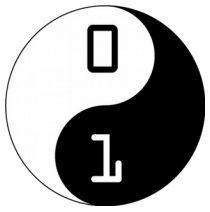
- Modifique a classe servidor para que:
 - no método **acessadoAgora** o atributo `lastAccess` com o horário atual.
 - o método **getTimeSinceLastAccess** retorne quanto tempo (em milisegundos) se passou desde o último acesso ao servidor
 - o método **isAccessible** retorne **true** caso o tempo que se passou desde o último acesso seja maior que a constante **ACESSO_MILIS** e retornará **false**, caso contrário
- Use o método `System.currentTimeMillis()` para saber o horário atual



Coding Dojo – Escalonador



- Cada servidor possuir uma fila de páginas.
 - Crie um Map (LinkedHashMap) em que a chave é o servidor e, o valor, é uma fila de páginas
 - O LinkedHashMap preserva a ordem de inserção
- Deverá existir uma fila de servidores
- As filas podem ser implementadas como ArrayList ou LinkedList



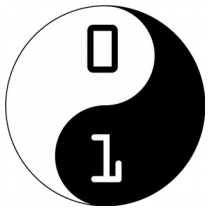
Coding Dojo – Escalonador

Método: adicionaNovaPagina

Este método adiciona uma nova URL na fila, retornando **true**, caso tenha adicionado com sucesso.

- Este método retorna **false** caso não seja possível adicionar esta página, ou seja:
 - Se página já foi adicionada (alguma hora) na fila
 - Precisamos de mais alguma estrutura para isso!
 - Se profundidade esteja maior que o previsto
 - A classe **URLAddress** possui o atributo **depth** para isso
 - Crie uma constante na classe EscalonadorSimples para indicar o limite de profundidade
- Ao adicionar a URL na fila:
 - Caso este servidor não exista, o mesmo deve ser criado para antes de adicionar a esta URL
 - Use os metodos do Map: `containsKey(key)`, `put(key,val)` e `get(key)`
- Você irá precisar de utilizar o método `getAddress()` da URL pois ele retorna o endereço URL (sem parâmetros):

<http://www.google.com/?q=asdkasodk>



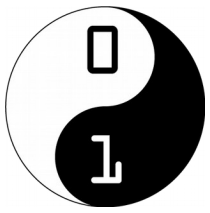
Coding Dojo – Escalonador

Método: getURL

Este método retorna a próxima URL na fila, removendo-a da mesma. Caso não haja URL disponível, a thread fica em modo de espera por 1 segundo (usando o método wait).

Para isso, faça o seguinte:

- Procure o primeiro domínio acessível e que a fila não esteja vazia. Caso encontre:
 - Extraia o primeiro elemento da fila (removendo-o).
 - Invoque o método **acessadoAgora** deste servidor para indicar que ele foi acessado
 - Retorne esta URL
- Caso não encontre URL, ponha a thread para esperar 1 segundo.



Coding Dojo – Escalonador Gerenciamento do Robots.txt

Exemplo de uso da API JrobotX:

```
RobotExclusion robotExclusion = new RobotExclusion();

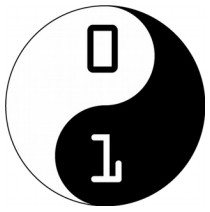
Record rFB = robotExclusion.get(new URL("https://www.facebook.com/robots.txt"), "daniBot");
System.out.println("Aceitou o fb index? " + rFB.allows("/index.html"));
System.out.println("Aceitou o fb o cgi-bin? " + rFB.allows("/cgi-bin/oioi"));
System.out.println("Aceitou o fb o oioi? " + rFB.allows("/lala/oioi"));

Record rTerra = robotExclusion.get(new URL("http://www.terra.com.br/robots.txt"), "daniBot");
System.out.println("Aceitou o terra index? " + rTerra.allows("/index.html"));
System.out.println("Aceitou o terra o cgi-bin? " + rTerra.allows("/cgi-bin/oioi"));
System.out.println("Aceitou o terra o oioi? " + rTerra.allows("/lala/oioi"));
```

O método **get** (sublinhado) fará a requisição ao robots.txt e retornará uma instancia do tipo Record.

Como a Requisição tem um custo alto, quem deve fazê-la é o **PageFetcher**.

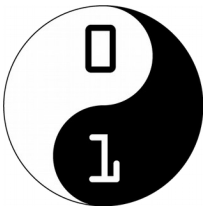
Mas, as instancias de Record por servidor devem ser armazenadas de forma centralizada (no **Escalonador**)



Coding Dojo – Escalonador Gerenciamento do Robots.txt

No escalonador, você deverá criar:

- Um atributo do HashMap para mapear cada servidor (chave) ao seu respectivo Record (valor)
- O método getRecordAllowRobots que retorna o Record de um determinado domínio de uma URL (passada como parâmetro)
 - Retorna null caso não exista
- O método putRecorded (threadsafe) que adiciona uma instância Record referente a um determinado domínio

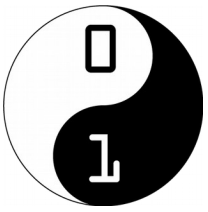


Coding Dojo – Escalonador

Contabilizar páginas e Finalizar coleta

A coleta será finalizada após serem coletadas 1.000 páginas (inicialmente, você pode diminuir o número máximo, para testar). Para que isso seja possível crie:

- um atributo na classe `EscalonadorSimples` que será o contador de páginas coletadas
- o método **`CountFetchedPage`** que incrementa o contador de páginas coletadas. O `PageFetcher` que deverá executar este método sempre que colete com sucesso uma página
- o método **`finalizouColeta`** que retorna verdadeiro caso já tenha coletado o limite de páginas a coletar.
 - crie a constante que indica o limite de páginas a coletar



Coding Dojo – Escalonador

Melhoria - LinkedHashMap

Para a fila, ao invés de usar o HashMap, utilize o LinkedHashMap

Assim, será preservado a ordem de inserção ao navegar pelo hashmap.

Sempre que não exista URLs em um servidor, você deverá excluir o servidor da fila.