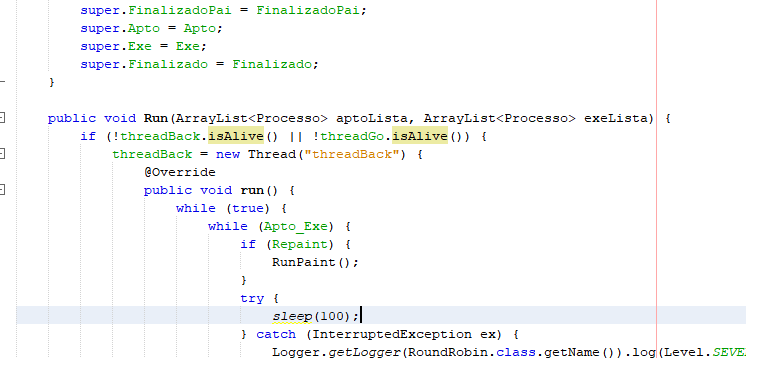
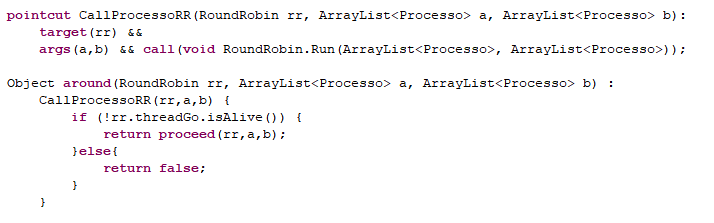
1. Remover a verificação sobre as threads antes de iniciar o método -> “if (!threadBack.isAlive() || !threadGo.isAlive()) : “Para remover os futuros erros de concorrência sem “apelar” para o synchronized usa-se esse conceito de verificação e futuramente poderá usar-se no método main alguma verificação se as threads estão sendo usadas, e esse aspecto ajuda a redefinir essa entrada” :





* pointcut CallProcessoRR(RoundRobin rr, ArrayList<Processo> a, ArrayList<Processo> b):

target(rr) &&

args(a,b) && call(void RoundRobin.Run (ArrayList<Processo>, ArrayList<Processo>));

Object around(RoundRobin rr, ArrayList<Processo> a, ArrayList<Processo> b) :

CallProcessoRR(rr,a,b) {

if (!rr.threadGo.isAlive()) {

return proceed(rr,a,b);

}else{

return false;

}

}

* pointcut CallProcessoSJF(SJF sjf, ArrayList<Processo> a, ArrayList<Processo> b):

target(sjf) &&

args(a,b) && call(void SJF.Run(ArrayList<Processo>,ArrayList<Processo>));

Object around(SJF sjf, ArrayList<Processo> a, ArrayList<Processo> b) :

CallProcessoSJF(sjf,a,b) {

if (!sjf.threadGo.isAlive()) {

return proceed(sjf,a,b);

}else{

return false;

}

}

* pointcut CallProcessoLTG(LTG ltg, ArrayList<Processo> a, ArrayList<Processo> b):

target(ltg) &&

args(a,b) && call(void LTG.Run(ArrayList<Processo>,ArrayList<Processo>));

Object around(LTG ltg, ArrayList<Processo> a, ArrayList<Processo> b) :

CallProcessoLTG(ltg,a,b) {

if (!ltg.threadGo.isAlive()) {

return proceed(ltg,a,b);

}else{

return false;

}

}

2. Exibe todas as informações do processo no console toda vez ao se criar um novo processo : “Não há nenhum método que exibe as informações ao criar um processo, então implementar o processo ajuda a verificar os resultados possibilitando a criação de um gráfico dos processos criados, outro ponto seria que o processo já está cheio de métodos e mais um para exibir as informações ocupará mais espaço”;

