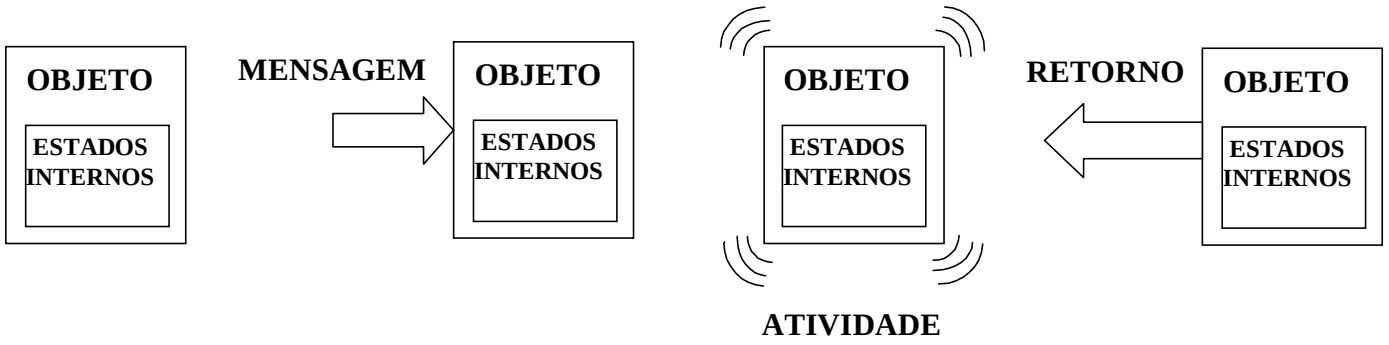
Agentes e Objetos

- Objetos (em software)
  - Metáforas computacionais que emulam características dos objetos do cotidiano
    - possuem estados internos que o descrevem
    - podem receber mensagens, a partir das quais realizam uma atividade, que é determinada conforme as mensagens enviadas e seus correspondentes parâmetros
    - podem ser classificados hierarquicamente em classes, que basicamente descrevem os estados internos e as mensagens que um objeto de uma determinada classe pode receber

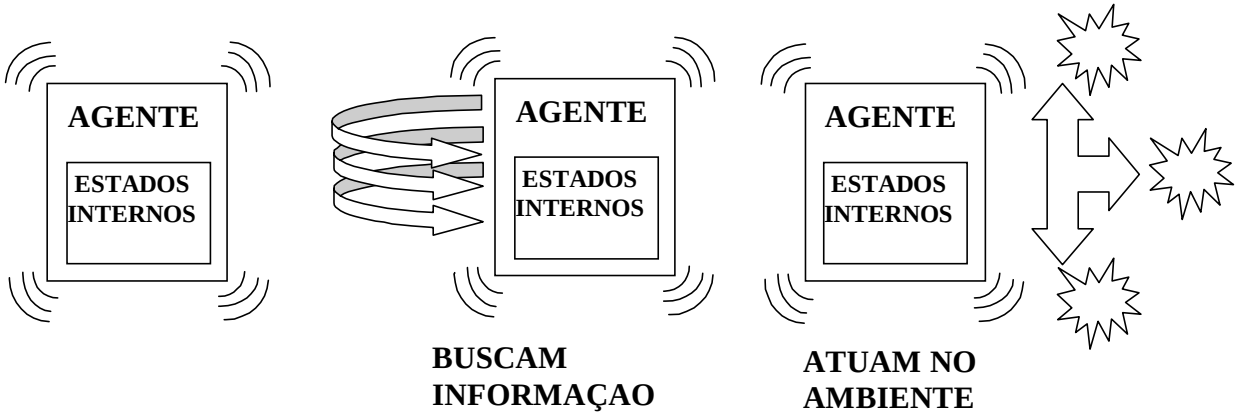


Agentes e Objetos

- Objetos = Máquinas
  - alimenta com mensagens
  - obtém o comportamento prescrito pela mensagem
- Características
  - previsibilidade do comportamento do objeto
  - cada mensagem = comportamento desejado
  - objeto não age por si só
    - responde a uma requisição de serviço
  - enquanto não está "em serviço"
    - inerte
  - objeto não "busca" mensagens ... só as recebe

Agentes e Objetos

- Agentes (em software)
  - metáforas computacionais que emulam comportamento de agentes do cotidiano
    - possuem estados internos que o descrevem
    - podem extrair dados de seu ambiente por meio de seus sensores e atuar sobre o ambiente por meio de seus atuadores
    - possuem um ciclo de vida interna por meio do qual sensoreiam e atuam

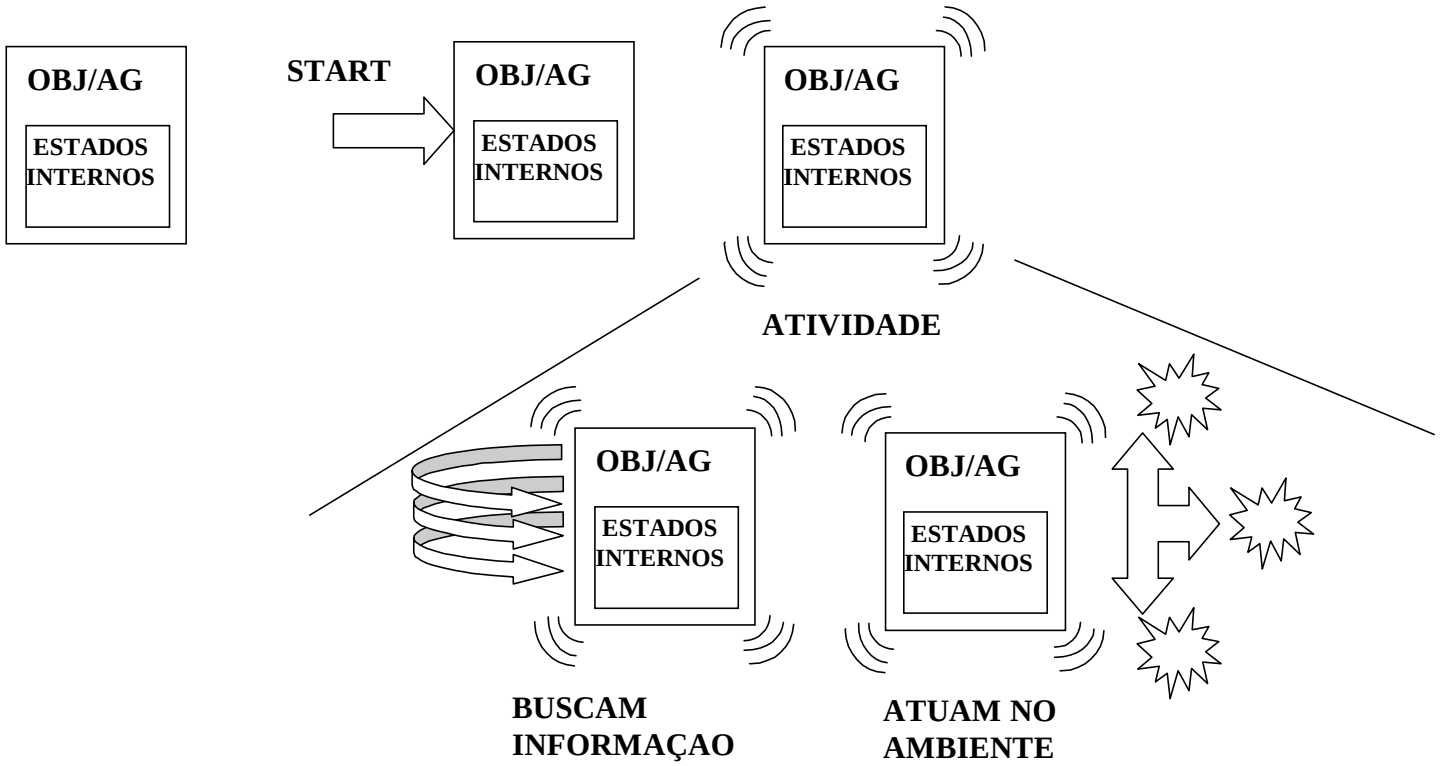


Agentes e Objetos

- Agentes = Organismos
  - mantém atividade incessante de busca por informação e atuação no ambiente
  - comportamento é determinado pela constituição do agente
- Características
  - nunca cessa atividade
  - comportamento pode ser imprevisível
  - agente age por si só
  - agente não "recebe comandos", mas busca por mensagens enviadas na forma de informações do ambiente, que são decodificadas, podendo influenciar o comportamento do agente

Agentes e Objetos

- Objetos podem ser agentes ?



Agentes e Objetos

- Objetos-Agentes
  - são criados como objetos normais de uma linguagem
    - podem criar objetos-filhos para seu próprio uso
  - são "startados"
  - permanecem em atividade até que "morram", por vontade própria
  - também podem criar agentes-filhos, com os quais mantém algum tipo de contato
- Problema
  - Um método que nunca termina paralisa o sistema
- Solução
  - Técnicas de programação concorrente

Processos e Threads

- Time-sharing
  - mecanismo pelo qual um certo número de "processos" que se deseja que sejam executados concorrentemente tenham acesso a "fatias" do tempo do processador, sendo interrompidos assim que estas "fatias" se esgotam e retomando sua execução quando lhe são atribuídas novas "fatias" de tempo
- Cada Processo
  - possui sua própria porção de memória
  - executado independentemente de outros processos
- Recursos comuns
  - compartilhados via mecanismos de sincronização
    - semáforos, filas, monitores, etc.

Processos e Threads

- Comunicação Inter-processos
  - variáveis globais não protegidas
  - variáveis compartilhadas protegidas por monitores
  - passagem de mensagens via recursos do sistema operacional
- Threads
  - trechos de código, dentro de um mesmo processo, que se deseja que rodem concorrentemente
  - grânulos finos concorrentes de um mesmo processo
  - compartilham todos os recursos do processo do qual são originados
  - são criados dentro de um processo e a ele pertencem
  - dependem da disponibilidade do sistema operacional
  - ou ... podem ser emulados por meio de um processo que os gerencie

Threads e Java

- Algumas linguagens de programação
  - contém diretivas para o uso de threads
- Java
  - contém toda a infra-estrutura para a programação multi-thread
  - interpretador Java emula os threads dentro do processo ao qual pertence
  - Suporte à programação multi-thread é centralizado em torno da classe java.lang.Thread
- Classe Thread
  - permite a criação de objetos do tipo Thread, onde cada objeto terá seu próprio fluxo de controle independente

Threads e Java

- Dois mecanismos para a criação de Threads
  - Mecanismo 1:
    - cria-se uma sub-classe da classe Thread
    - efetua-se um override to método run(), que é o ponto de entrada para a execução do thread
    - após a criação de um objeto desta classe, invoca-se o método start(), que inicia o novo thread e por sua vez invoca o método run()
  - Mecanismo 2:
    - cria-se uma classe implementando a interface Runnable
    - implementa-se o método run() da interface Runnable
    - cria-se um objeto da classe Thread, passando para o construtor um objeto da classe implementando Runnable
    - invoca-se o método start() do objeto do tipo Thread

Threads e Java

- Estados de um Thread
  - new thread - assim que é criado
  - runnable - depois que é startado
  - not runnable - bloqueado esperando por um evento
  - dead - quando o método run() retorna
- Métodos Importantes
  - sleep() - faz o thread "dormir" por um certo tempo
  - interrupt() - interrompe definitivamente a execução do thread
  - join() - bloqueia o thread corrente até que o thread indicado morra
  - is_alive() - verifica se um thread está ainda "vivo"

Threads e Java

- Prioridades
  - set_Priority(), get_Priority()
  - MIN_PRIORITY(1), NORM_PRIORITY(5), MAX_PRIORITY(10)
- Sincronização
  - Métodos Sincronizados
    - synchronized function(...)
  - Statements Sincronizados
    - synchronized (expression) statement
- Monitores e Sincronização de Processos
  - wait()
  - notify()
  - notifyAll()

Threads e Java

- Threads do tipo Daemon
  - Threads que executam em background e provêm serviços a outras threads ou processos
  - normalmente executam continuamente um conjunto de instruções onde esperam por uma requisição de serviço, realizam o serviço solicitado e passam a aguardar nova requisição de serviço
  - podem ser utilizadas para a criação de agentes que passam a ter vida própria, ficando independentes do processo que as originou
- Em Java
  - método setDaemon() deve ser invocado antes de se dar o start() da thread

Agentes e Redes

- Redes de Computadores
  - Nível superior de concorrência que permite o desenvolvimento de aplicações em paralelo
- Modelos de Comunicação Entre Processos
  - Sockets (Passagem de Mensagens)
  - RPC - Remote Procedure Call
  - Objetos Distribuídos (CORBA e DCOM)
- Passagem de Mensagens
  - mecanismo mais simples para comunicação entre/com agentes
- Java
  - provê mecanismos para os três tipos de modelo de comunicação entre processos
    - Sockets, RMI, CORBA

Meu Primeiro Agente (EC1)

- Agente de Mirror
  - dados dois diretórios designados em um computador, o agente de mirror deve se encarregar em manter ambos os diretórios exatamente iguais
  - havendo uma inserção de arquivo ou subdiretório em um dos diretórios, o agente deve inserí-lo no seu diretório mirror
  - havendo modificação nos arquivos de um dos diretórios, o agente de mirror deve copiar os arquivos modificados para o diretório mirror (utilizando a data do arquivo para definir qual é o mais atual)
  - apagando-se um arquivo ou subdiretório de um dos diretórios, ele deve ser apagado também no diretório mirror

Meu Primeiro Agente (EC1)

- Requisitos do Agente de Mirror
  - O agente de Mirror deve funcionar como um daemon
  - Os diretórios que devem ser espelhados devem ser passados ao agente como um parâmetro (não devem ser embutidos no código)
  - Havendo um arquivo com o nome KILL_AG.$$$ em um dos diretórios mirror, o agente deve se matar
  - O agente somente procederá a mudanças nos diretórios quando os arquivos estiverem fechados ... ou seja, enquanto algum arquivo estiver aberto, ele não procederá a nenhuma mudança referente ao arquivo em questão
  - o agente deve utilizar algum mecanismo que evite o consumo de muito tempo de CPU, para não entrar em conflito com outros processos que estejam rodando na mesma máquina

Meu Primeiro Agente (EC1)

- Perguntas a serem respondidas no relatório
  - Qual é o ambiente deste agente ?
  - Quais são os sensores deste agente ?
  - Quais são os atuadores deste agente ?
  - Como é o ciclo de vida deste agente ?
  - Qual a melhor classificação, dentre as estudadas no curso, para este agente ? (Reflexivo, Comportamental, Planejador, Emocional, Comunicativo ou Semiótico ?)
  - Qual a aplicação deste tipo de agente ? (Robótico, Desktop, Internet, Entretenimento, etc ?)
  - Podemos dizer que este agente é inteligente ? Por que ?