Das Ask-Reply-Konzept der WBS-Agenten

Definierbare Schnittstellenfunktionen

```
Konstruktor
Destruktor
virtual void AgentMainLoop ();
virtual void AgentConfig (const BO_String &slot, const BO_String &value);
virtual void AgentInit ();
virtual void AgentReset ();
virtual void AgentConnectionHandler (const BO_String &agentname);
virtual void AgentDisconnectionHandler (const BO_String &agentname);
virtual void AgentTellMessageHandler (Message *msg);
virtual void AgentAskMessageHandler (Message *msg);
virtual void AgentReplyMessageHandler (Message *msg);
```

Aufbau von Messages

Da alle Messagetypen von der Basisklasse `Message` abgeleitet sind, bieten sie alle den gleichen Grundaufbau. Z.B. besitzt jede Messageklasse zu ihrer Identifikation eine eindeutige Messagetyp-ID. Die Methoden, die alle Messages gemeinsam haben, sind:

`const BO_String& getID ();`
Nach dem Empfang einer Message kann über diese Methode die ID des Messagetyps, von der diese Message eine Instanz ist abgefragt werden.

`const BO_String& getSenderName ();`
Über diese Methode kann der Name des absendenden Agenten ermittelt werden.

`void autoDeleteProtect ();`
Nach dem Verlassen der Message-Handler-Funktion wird die Message normalerweise automatisch aus dem Speicher entfernt. Soll sie aus irgendwelchen Gründen noch aufgehoben werden, so läßt sich über diese Methode die Message vor dem Löschen schützen. Es muß aber anschließend selbst für das Entfernen der Message aus dem Speicher gesorgt werden!

`int isAutoDeleteProtected ();`
Über diese Funktion läßt sich abfragen, ob die Message vor dem automatischen Löschen geschützt ist.

`int tell (const BO_String &addressee);`
Mit dieser Methode wird die Message an den Agenten mit dem angegebenen Namen gesendet.

`int tellAll ();`
Mit dieser Methode wird die Message an alle anderen Agenten gesendet.

`int ask (const BO_String &addressee, BO_String &ID);`
Mit dieser Methode wird die Message als Anfrage an den angegebenen Agenten gesendet. Der Message wird eine eindeutige ID zugeordnet, die in dem String ‚`ID`' zurückgegeben wird.

`int askAll (BO_String &ID);`
Mit dieser Methode wird die Message an alle anderen Agenten als Anfrage gesendet. Wie bei `ask` erhält auch diese Message eine ID.

`int replyTo (Message *msg);`
Mit dieser Methode antwortet man auf eine Anfrage-Message.

`const BO_String& getReplyMessageID ();`
Mit dieser Methode erfragt man die Message-ID einer ‚replyten' Message. Diese korrespondiert zu der ID, die beim Ask vergeben wurde.

Aufgabe 1: Der Goldsammler

Eine Welt voller Goldklumpen wartet auf euch. Programmiert einen Agenten, der in der Welt herumläuft und möglichst viel Gold einsammelt. Die Auswahl einer „Strategie" bleibt euch überlassen.

Aufgabe 2: Der Blindenhund

Ein Agent, nennen wir ihn „Sammler", möchte - wie schon viele andere Agenten vor ihm - möglichst viel Gold einheimsen. Leider hat er ein Problem: Aufgrund von Nachlässigkeiten der Schöpfer funktioniert seine visuelle Wahrnehmung nicht mehr. Er behilft sich indem er einen zuverlässigen Partner, nennen wir ihn „Sucher", anheuert, der glücklicherweise nicht das geringste Interesse an Gold hat.

Programmiert jeweils einen „Sammler" und einen „Sucher". Der Sammler darf die visuelle Information des Welt-Servers nicht verarbeiten, der Sucher darf kein Gold aufnehmen. Der Sammler soll den Sucher regelmäßig per „ask" nach dem nächsten Schritt fragen. Der Sucher antwortet darauf mit „replyTo".

Das Nachrichtenformat lautet:
ask: „WOHIN"
replyTo: „VORNE", „HINTEN", „RECHTS" oder „LINKS"

Achtet zuerst darauf, daß der Nachrichtenaustausch funktioniert und der Sammler entsprechende Anweisungen des Suchers ausführt. Welche Strategie der Sucher einsetzt bleibt wieder euch überlassen.