# エージェントベースの取引所理論

## ～実験編～

平成 16 年 1 月 30 日


指導教官：　有賀裕二

学籍番号：　00C2235011L

氏名：　　中田翔

# 目次

# 第 1 章 序論

## 1.1 背景と目的

このゼミに入ることを決意した当時，私は公認会計士受験のために経理研究所で簿記漬けの毎日を送っていた。朝 6 時に家を発ち，夜 12 時に帰宅する生活を続ける中，日に日に募る疲労，手応えのない模試の結果に行き詰まりを感じ始めていたとき，このゼミの存在を知った。

「仮想市場実験」,「コンピュータ」，といったテーマを掲げるこのゼミに，他のゼミにはない“ 新鮮さ“ を感じた私は，簿記から離れることで何か違ったものが見えてくるかもしれない，という淡い予感を感じ，入ゼミを決めた。

入ってからも苦労の連続であった。特にプログラミングを伴なう作業に，PC に触れたことがある程度の我々が挑むのは，中々大変なことであった。しかし，未知の分野に取り組み，多少なりとも結果を出すことで得たものも大きいと感じている。この論文は，そんな我々の 2 年間の集大成と言うべきものになっている。

## 1.2 方法と意義

この卒業論文は，理論，実験，解析の三部構成となっている。各ゼミ生が一部ずつ担当しており，このうち私は実験編を担当する。

我々は，テーマの一つである仮想市場実験に，U-Mart というツールを使用している。U-Mart は，人間とコンピュータが混在可能な仮想的な市場取り引きの実験システムである。理論から実践と，三部通して読むことで，我々の最大のテーマである，U-Mart を使った学習の下地作りとなるような構成になっている。読者諸氏も是非役立てて頂きたいと思う。

## 1.3 論文の構成

実験編は，U-Mart キットの具体的な使い方から，Java を使ったマシンエージェントのプログラミング，ネットワーク版 U-Mart のサーバの構築から実際の実験へと，次に続く解析編への足がかり的な内容となっている。文系の学生にとって，ネックになると思われるのがプログラミングであるが，そこは実際のマシンエージェント作成に必要な最低限の知識を，出来るだけ分かりやすく記したので，初学者も恐れず，挑戦していただきたいと思う。

# 第 2 章　U-Mart プロジェクトについて

## 2.1　はじめに

近年，人工市場という研究テーマが経済学者はもちろんのこと，工学者，数理科学者等からも注目を集めている。U-Mart 研究会は人工市場を媒体とした，経済学者と工学者の交流の場を提供することを目的として設立された。学際的な教育・研究活動を通じて，理論，実験に続く，社会科学における新しい第 3 モードの研究様式を目指している。

U-Mart 組織委員会ではこれまでに，株価指数を取り引きする仮想先物市場シミュレータを開発し，進化経済学会等でのデモンストレーション，また関西－関東－北海道を結んだ遠隔取引実験等を行ってきた。

## 2.2 U-Mart プロジェクトの歩み

U-Mart プロジェクトはこれまで，様々な活動を行ってきた。以下に主要な活動を年表形式で記す。

| | |
|---|---|
| 1998 年 | 「複雑系夜話　進化する経済と経済学」（招待講演）<br>塩沢由典，第4回創発システムシンポジウム SICE 夏の学校　かずさアカデミアパーク<br>招待公演とその後の談話で人工市場の構想が生まれる。 |
| 1999 年 | 3 月 27 日　進化経済学会大阪大会<br>プロポーザルセッション<br>「進化する経済の実験室創生のために-バーチャル市場の参加型シミュレーション-」<br>出口弘 「進化する経済の実験室創生のために -バーチャル市場の参加型シミュレーションの意義-」<br>塩沢由典 「V-Mart の意義　共通テストベッドとしてのバーチャル市場 -経済学のロボカップを目指して- 」<br>喜多一　「プログラムの進化」<br>寺野隆雄 「エージェントは計算しないとわからない！？」<br>生田目章　「制度と場のメカニズムの工学的な観点からの解明」 |
| 2000 年 | 3 月 25-26 日　進化経済学会東京大会<br>（U-Mart の初期バージョンが完成，デモンストレーションが行われる） |
| 2000 年 | 8 月　創発システムシンポジウムで，最初の公開実験「PreU-Mart2000」が開催される |
| 2001 年 | 5 月 21 日　AESCS<br>H. Sato, H. Matsui, I. Ono, H. Kita, T. Terano, H. Deguchi, Y. Shiozawa<br>"U-Mart Project Learning Economic Principles<br>from The Bottom by Both Human and Software Agent" |
| 2001 年 | 7 月 8 日　CASOS2001<br>Hiroyuki MATSUI, Hiroshi SATO, Takao TERANO, Yoshinori SHIOZAWA, |

| | |
|---|---|
| | Isao Ono, Hiroshi Deguchi, Hajime Kita:<br>"Learning Economic Principles from the bottom<br>by both human and Software Agents - Outline of U-Mart Project" |
| 2001 年 | 8 月 24-26 日 SICE 夏の学校 U-Mart2001 (25 日)<br>喜多一 「U－Mart の教育への展開」<br>松井啓之 「ゲーミングシミュレーターとしての U-Mart」<br>佐藤浩 「U-Mart2001 仮想先物取引コンテスト」<br>佐藤浩, 小野功, 高尾頼和 「U-Mart 次期サーバについて」<br>喜多一 「U-Mart 入門」 |
| 2001 年 | 8 月 24-26 日 SICE 第 7 回 創発システムシンポジウム<br>湯浅秀男, 喜多一, 稲垣伸吉:U-Mart によるプロジェクト演習 |
| 2001 年 | 1 月 26-27 日 SICE 第 13 回 自律分散システム・シンポジウム<br>佐藤浩, 松井啓之, 小野功, 喜多一, 寺野隆雄<br>オープン型人工市場におけるエージェントの戦略と市場の挙動 (Pre U-Mart 2000 実施報告) |
| 2001 年 | 3 月 30-31 日 進化経済学会福岡大会<br>寺野隆雄 「U-Mart 研究プログラムの概要」<br>佐藤浩 「ソフトエージェントによる U-Mart PreU-Mart2000 実施報告」<br>出口弘, 松井啓之 「ゲーミングシミュレーターとしての U－Mart」<br>塩沢由典 「U-Mart 経済学として得られた事」<br>谷口和久 「教育キットとしての U－Mart」<br>喜多一 「U-Mart の工学分野での研究, 教育への可能性」 |
| 2001 年 | 5 月 17 日 U-Mart 講習会 (大阪市立大学)<br>小山友介 「先物取引について」<br>谷口和久 「ヒューマン・エージェントと基本的戦略」<br>中島義裕 「マシン・エージェントのアルゴリズム」<br>橋本文彦 「マシン・エイド・ヒューマン・エージェントの利用」<br>松井啓之 「東工大版エージェント開発キット」<br>佐藤浩 「サーバー・マシンのインストールと環境」 |
| 2002 年 | 1 月 7-8 日 修善寺合宿<br>田中美栄子 「経済物理学の過去, 現在, 未来」<br>中島義裕 「U-Mart を用いたこれからの研究計画」<br>高尾頼和 「遺伝的アルゴリズムによるエージェントベースシミュレーションに基づく近似モデルの構築」<br>川辺 「人工市場を用いたエージェントの利益最大化」<br>粂井淳子 「マーケットシステム －U-Mart 市場におけるマーケットメーカー導入の検討」<br>小山友介 「U-Mart エージェントの分類」 |

| | |
|---|---|
| 2002 年 | 3 月 30 日 進化経済学会大阪千里山大会<br>寺野隆雄 「U-Mart プロジェクト：人工市場研究から制度設計へ」<br>中島義裕 「U-Mart を用いたこれからの研究計画」<br>谷口和久，松井啓之，出口弘，五十嵐寧史 「教育ツールとしての U-Mart：経済学教育での活用事例」<br>喜多一，湯浅秀男 「教育ツールとしての U-Mart：工学教育での活用事例」<br>佐藤浩，小山友介 「U-Mart2001 実験報告および既存エージェントの分類」<br>高尾頼和，小野功，小野典彦 「エージェントベースドシミュレーションに基づく近似モデルの進化的構築」<br>植木潤吾，森直樹，甲斐啓仁，深瀬澄，佐藤浩，後藤岳，上田智巳，粂井淳子，中島義裕<br>「U-Mart によるシミュレーション研究」 |
| 2002 年 | 6 月 22 日 CASOS2002 (UMIE2002)<br>Terano Takao, Deguchi Hiroshi, Kita Hajime, Shiozawa Yoshinori,<br>Robert Axtell, Kathleen Carly, Maksim Tsvetovat, Sato Hiroshi, Matsui Hiroyuki Ono Isao<br>"UMIE-2002: U-Mart International Experiment 2002<br>-What we have learned from the virtural market-" |
| 2002 年 | 8 月 18 日 SICE 夏の学校<br>UMIE-2002:U-MART 国際実験報告("UMIE-2002: What we have learnt from the U-Mart Project")<br>寺野隆雄，松井啓之，佐藤浩，小野功，出口弘，喜多一，中島義裕，塩澤由典 |
| 2002 年 | 9 月 24 日－28 日 日本機械学会 2002 年次大会<br>湯浅秀男，喜多一，小林智彦：プロジェクト演習による教育実例 U-Mart による 教育ツール |
| 2002 年 | 11 月 3 日 計測自動制御学会 第 26 回システム工学部会研究会「人工市場研究の 現状と展開」<br>森，小野，松井，佐藤，喜多，出口：U-Mart システムとオブジェクト指向設計・プログラミング ---<br>U-Mart サマースクール実施報告 , U-Mart の情報系教育への利用 |
| 2002 年 | 11 月 4 日 U-Mart2002 (大阪市立大学)<br>松井啓之 「U-Mart 実験概要」<br>中島義裕 「U-Mart2002 実験方法」<br>松井啓之 「UMIE2002 実験報告と今後」 |
| 2003 年 | 3 月 29 日 進化経済学会第 7 回東京 (専修大) 大会<br>小野功，佐藤浩，森直樹，喜多一，松井啓之<br>「社会経済シミュレーションのためのソフトウェア開発方法論」予稿集 pp. 283-288<br>喜多一 「エージェントベースドモデルによる社会システムの理解」予稿集 pp. 435-439<br>谷口和久 「ゲーミングによる価格形成メカニズムの理解<br>-コースウェアとしての U-Mart」予稿集 pp. 289-292<br>塩澤由典 「U-Mart から生まれる新しい経済学」 |
| 2003 年 | 6 月 20 日-25 日 NAACSOS2003 (UMIE2003)<br>North American Association for Computational Social and Organizational Science Conference |

| | |
|---|---|
| | Hiroshi Deguchi,Takao Terano,Hajime Kita,Yoshinori Shiozawa,Robert Axtell,Katheleen Carley,<br>Maksim Tsvetovat,Hiroshi Sato,Hiroyuki Matsui,Isao Ono,Yoshihiro Nakajima,Naoki Mori<br>(Hajime KITA)"U-Mart International Experiment2003(UMIE2003)"<br>(Yusuke KOYAMA),"Report of UMIE2002 -Strategy and Rank Order of Submitted Machine Agents"<br>Yoshihiro NAKAJIMA, Tomomi UEDA, "Analysis of Submitted Agent to UMIE2002<br>- Influence of Spot Data and Oppornents on Agents-" |
| 2003 年 | 7 月 16 日-19 日 CIRA2003<br>IEEE Computational Intelligence in Robotics and Automation<br>KAZUSA AKADEMIA Park<br>Hajime KITA, Hiroshi SATO, Naoki MORI, Isao ONO,<br>"U-Mart System, Software for Open Experimnets of Artificial Market"<br>Tomomi UEDA, Kazuhisa TANIGUCHI, Yoshihiro NAKAJIMA,<br>"An Analysis of U-Mart Experiments by Machine and Human Agents" |
| 2003 年 | 8 月 25 日-8 月 29 日<br>ISAGA2003 International Simulation And Gaming Association (ISAGA)<br>The 34th Annual Conference Hosted by Science Council of Japan (SCJ)<br>and Japan Simulation And Gaming Association (JASAG)<br>8 月 27 日 ISAGA U-Mart2003<br>8 月 28 日 ISAGA U-Mart セッション<br>8 月 29 日 ISAGA Agent Based Modeling Meets Gaming Simulation 5 |

## 2.3 共通テストベッドとしての U-Mart

### 2.3.1 U-Mart 標準キットの説明

U-Mart は，コンピュータ上のエージェントプログラムと実際の人間の実験参加者とが混在する状況で取引きが行われる仮想的な市場取引きの実験システムである。これは，経済学，システム工学，計算機科学，複雑適応系，人工知能の実験的研究のための共通テストベッドを提供することを目的としている。現実の株価指数に対する架空の先物を取引き対象とすることで，独自の価格形成機能を持ち，かつ，現実とリンクする実験空間を実現する。

- 人間とプログラムの混在
- 先物取引きにより現実とのリンクの存在
- 制度構造の解析を可能とする

以上３点が，U-Mart の大きな特長である。

U-Mart は，市場の動きと市場における経済行動を解明するために，模擬市場を設計し，動かしてみることによって，経済学の側面からは，

- 株式など売買行動に関する人間の判断様式が解明される。
- 市場の乱高下など，市場の投機的な動きを回避するための実験を行うことができる。

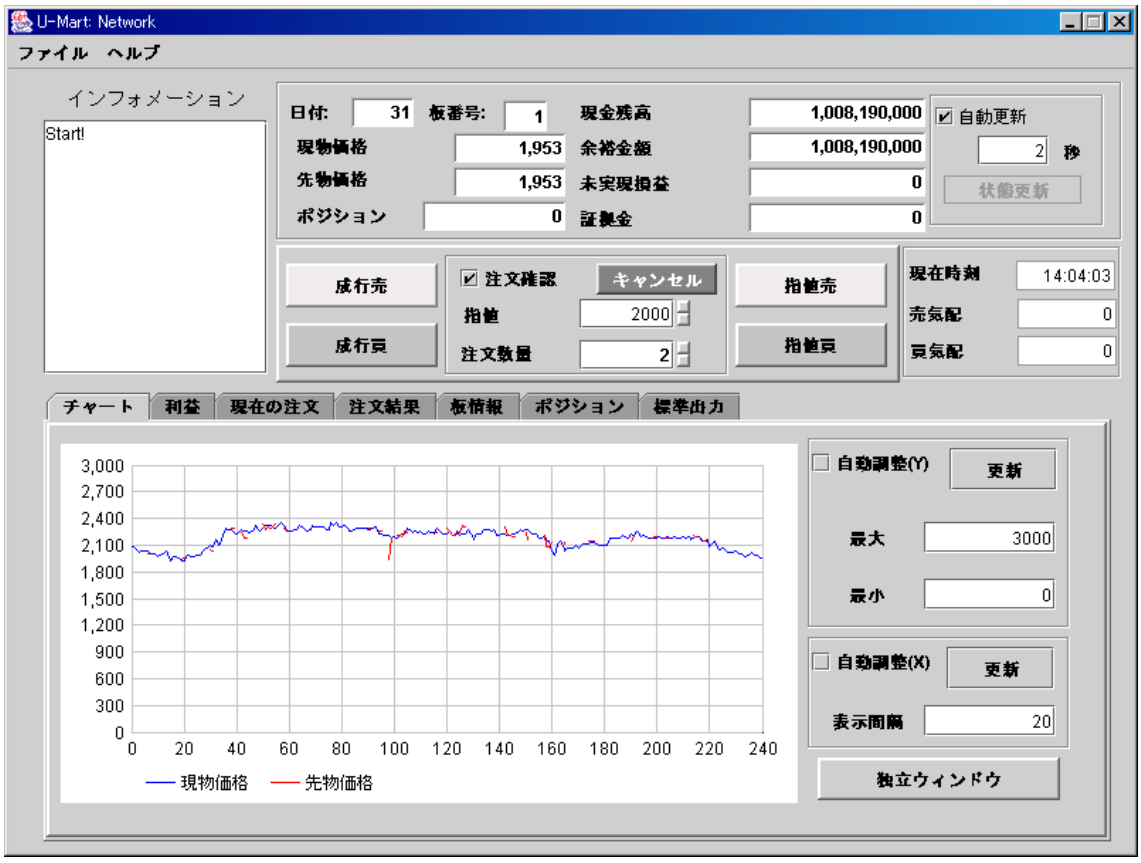
また，工学的な側面からは

- 市場に参加するマシンエージェントの製作に関連して，進化システム・適応学習・群知能など多様な分野への効果が期待できる。

以下，具体的にネットワーク実験用，エージェント開発用各キットの使い方について説明する。

#### 2.3.1.1 ネットワーク実験用キット

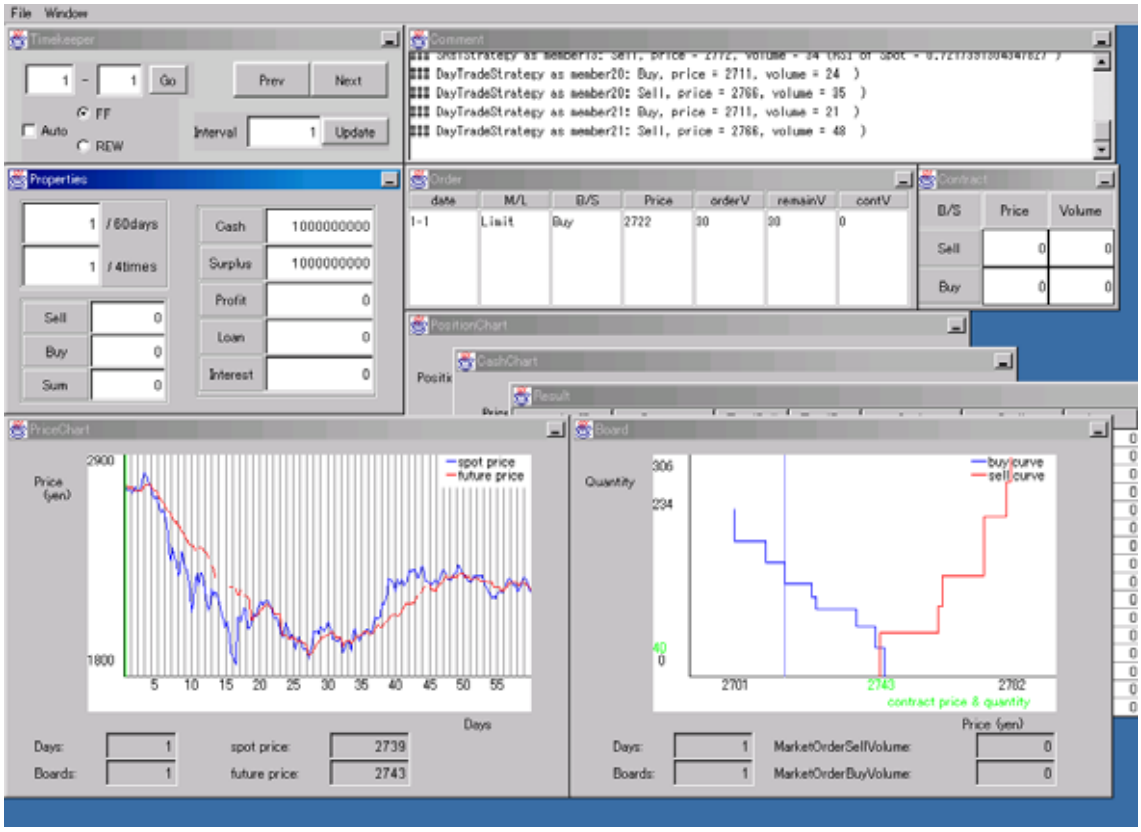
ネットワーク実験用キットのスクリーンショットを以下に示す。



売買は価格チャートや板情報等を参照しつつ，成行，指値で行えるので，実践的な取引を体験することができる。

#### 2.3.1.2 エージェント開発用キット

エージェント開発用キットのスクリーンショットを以下に示す。




基本的な操作はネットワーク実験用キットと同様である。価格チャートや板情報を参照しつつ，指値，または成行で注文する。

### 2.3.2 教育用ツールとしての利用

U-Martシステムは経済学，工学の両分野での教育に極めて効果的な教材となる。実際にU-Mart を用いた教育がいくつか実践されている。

経済学系(経済学，経営学，商学などの学部，大学院)の分野ではU-Mart は

- 市場が個々の取引エージェントの行動により形成されるという基本的な概念を実験的に示す
- 金融市場，とくに金融派生商品(デリバティブ)市場の考え方や制度を体験する

- 個々のエージェントの戦略や行動が市場に与える影響を示す

などの点で有効な教材である。利用方法としては，学生自身が公開実験などで提供されたエージェントとともに，市場取引に参加するゲーミング・シミュレーションが用いられる。また，ある程度，数式や計算機に馴染みさえあれば自らU-Mart システムを用いてソフトウェア・エージェントを構成することも社会科学系の学生でも十分に可能である。

一方，U-Mart は情報工学系の分野での教材としても有用である。

- 学部レベルの初級プログラミングの課題
- 大学院レベルでの予測，学習，適応，最適化やエージェントシミュレーションそのものの学習
- 工学系の学生に経済学や金融工学の紹介

などに有効である。その意味でU-MARTは機械系の学科などで見られるロボットコンテストと同様の効果を持っている。またエージェント開発は個人の単位で比較的短期間に取り組めるためカリキュラム編成上も扱いやすい。

### 2.4 おわりに

ここでは U-Mart プロジェクトについて，その歴史や実際のツールの使用法をみてきた。次章より Java プログラミングを利用したエージェントの実装について説明していく。

# 第 3 章　エージェントプログラムの実装

## 3.1 はじめに

U-Mart は，その目的の一つであるコンピュータプログラミングの教育の観点から

レベル 1：骨格となるプログラムを与えられて必要な部分を付け加える。
レベル 2：U-Martサーバとの通信用ライブラリだけ与えられて取引エージェント全体を自ら作成する。
レベル 3：プロトコル仕様書だけ与えられて通信部分も含めて全てのプログラムを自作する。
レベル 4：グラフィックインターフェイス (GUI)つきのプログラムや並行プログラミングなどの技術を利用したより高度なプログラミングを行う。U-Martシステム全体や種々の支援ツールをプログラミングの課題とする。

などの多様なレベルでの教育に利用できる。

以下にレベル１～2 へ挑むプログラミング初学者へ向けて，オブジェクト指向の概念も交えつつ，その導入部分を説明していく。

## 3.2 オブジェクト指向プログラミング

### 3.2.1 概要

オブジェクト指向プログラミングとは，データとそれを操作する手続きをオブジェクトと呼ばれるひとまとまりの単位として一体化し，オブジェクトの組み合わせとしてプログラムを記述するプログラミング技法である。プログラムの部分的な再利用がしやすくなるなどのメリットがある。

代表的なオブジェクト指向言語としては，C 言語にオブジェクト指向的な拡張を施した C++や，Sun Microsystems 社の Java，Xerox 社の SmallTalk，NeXT 社(現 Apple 社)が自社の OS である NeXT STEP 向けアプリケーションソフト開発用に開発した C 言語ベースの Objective-C などがある。

オブジェクト指向プログラミングを構成する概念には，次のようなものが挙げられる。

・カプセル化
・継承
・ポリモルフィズム

なお，この構成要件には異論・例外がある

### 3.2.2 カプセル化

カプセル化とは，データとそれを操作する手続きを一体化して「オブジェクト」として定義し，オブジェクト内の細かい仕様や構造を外部から隠蔽することである。

外部からは公開された手続きを利用することでしかデータを操作できないようにすることで，個々のオブジェクトの独立性が高まる。カプセル化を進めることによりオブジェクト内部の仕様変更が外部に影響しなくなり，ソフトウェアの保守性や開発効率が高まり，プログラムの部分的な再利用が容易になるといったメリットが得られる。

### 3.2.3 継承

オブジェクト指向プログラミングにおける継承とは，既に定義されているクラスをもとに，拡張や変更を加えた新しいクラスを定義することである。元になるクラスを「スーパークラス」(super　class), あるいは「基底クラス」「基本クラス」(base　class)などと呼び，新たに定義されたクラスを「サブクラス」(sub　class)，あるいは「派生クラス」(derived class)と呼ぶ。

スーパークラスの性質はすべてサブクラスに受け継がれ，サブクラスではスーパークラスとの違いを定義するだけでよい。複数のスーパークラスから新しいクラスを定義することを多重継承という。

### 3.2.4 ポリモルフィズム

ポリモルフィズムは，多相性，多義性，多態性とも呼ばれ，オブジェクト指向プログラミングにおいては，同じアクセスに対して，オブジェクトごとに違った振る舞いをすること（同一のメッセージを種々のオブジェクトが独自に解釈すること）を言う。

オブジェクトの型ではなく，生成されたオリジナルのクラス型で実装されたメソッドが起動するために，同じ型のオブジェクトに対して，同じメソッド呼び出しを実行しても，生成されたオリジナルのクラスが異なれば，異なるメソッドが起動される。

## 3.3 エージェントプログラムを書いてみよう

### 3.3.1 概要

U-Mart のゲーム環境では，ヒューマン・エージェント（人間）とマシン・エージェント（売買戦略アルゴリズム）が個々の戦略に基づいて注文を出す。各エージェントは，それぞれの目的に応じて戦略を立案しているため, ユーザは U-Mart を通じて多様な意思決定と，それによって生じる価格変動を観察することになる。

U-Mart において注文を執行するには, 以下の 3 つの変数に何らかの値を代入しなければならない。

い）　order.buysell:　　売りならば１，買いならば２，注文しないときは０を代入する

ろ） order.price: 注文価格を代入する
は） order.quant 注文数量を代入する

これらの変数に代入をするためのアルゴリズムが U-Mart における「売買戦略」としてマシン・エージェントに搭載されることになる。マシン・エージェントは，現物価格と先物価格の履歴を情報として受け取ることが可能であり，それらをもとに売買戦略を決定する。価格，出来高，時間を変数として市場を分析する方法は，テクニカル分析と呼ばれているが，U-Mart で参照可能な情報が価格，出来高，時間に加えて，ポジションと現在資産であることから，U-Mart における各エージェントの意思決定はテクニカル分析に近づく。

U-Mart のマシン・エージェントは Java により実装される。一般に，ある言語ではじめて組むアルゴリズムは“ Hello World ”という文字列を表示する簡易プログラムであり，この作業は慣習となっている。今回，この慣習に従い Java で“ Hello World ”を書くことから開始し，連続的なプロセスの中，最短距離でマシン・エージェントを開発する。作業は 7 つの Step に分かれており，Java の一般的な文法を学ぶだけではなく，クラスの抽出，メソッドの抽出といったオブジェクト指向プログラミングにとって重要な技術であるリファクタリング（ソースの再利用性を高める作業）を経験することになる。以下に 7 つの Step を紹介する。

Step 1) ”Hello World”を書く
Step 2) 売買判定結果，注文数量，注文価格を文字列で書く
Step 3) ローカル変数の導入
Step 4) フィールドの導入
Step 5) メソッドの抽出
Step 6) クラスの抽出
Step 7) 継承，getOrder( ) の実装

Step 7 が終了した段階で, 開発者は UMIE2003 標準キットで動作可能なエージェントがいつのまにか完成していることに気づくはずだ。

### 3.3.2 Step 1) ”Hello World”を書く

ここでは Java プログラムの作成方法を修得するために，最も簡単な Java プログラムを作成してみることにする。なお，プログラムの作成に最低限必要な環境は構築済みであることを前提に進める。

まずは Windows 付属のメモ帳などのテキストエディタを起動する。普段使い慣れているエディタがあるのなら，メモ帳の代わりにそちらを使っても構わない。インターネット上

には優れたエディタがいくつも公開されているので，自分に合ったものを探してみるのもいいだろう。その際はなるべく行番号を表示できるものが望ましい。ただし，Microsoft Word のようなワードプロセッサをエディタ代わりに使用するのはお勧めできない。

エディタを起動したら次の内容を入力してみる。Java プログラムは大文字と小文字を区別するので，入力間違いのないように注意しよう。

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("HelloWorld");
    }
}
```

入力が完了したらファイルに保存する。ここでは説明のため，「C:¥Java¥」以下に保存することにする。ファイル名は必ず「HelloWorld.java」とする。つまりプログラムの 1 行目がプログラムの名前であり，ファイルの名前にも対応している。末尾の「。java」は Java プログラムのソースであることを表している。ファイル名もやはり大文字と小文字が区別されているので注意が必要である。

人間が入力したプログラムは，コンピュータにとっては非常に理解しにくいプログラムなので，先程作成したプログラムをコンピュータが理解できる形に変更してやる必要がある。これをコンパイルという。

Java でのコンパイルは次の手順で行う。まずコマンドプロンプト起動する。すると通常「C:¥＞」と表示されているはずだ。これは C ドライブ直下にあるファイルを操作可能なことを意味している。今の場合，ファイルは「C:¥Java¥」の中に保存したので，作業場所を変更する必要がある。作業場所の変更には「cd」コマンドを使って，次のように入力する。表示内容が「C:¥Java＞」に変更されれば成功である。

```
C:¥>cd Java
```

続けてコンパイルを行ってみよう。Java でコンパイルするためのプログラム，すなわちコンパイラは「javac」という。このプログラムに，コンパイルしたいファイル名を教えると，コンパイルが実行される。ここでは次のように入力する。

```
C:¥Java>javac HelloWorld.java
```

画面に何もメッセージが表示されなければ成功だが，次のようなメッセージが表示された場合は残念ながら失敗である。この例のエラーメッセージは，4 行目（実際は 3 行目に書いたが）に；（セミコロン）が足りないことを示している。エラーが表示されなくなるまで，指摘された箇所の修正とコンパイルを繰り返し行ってみよう。

```
C:¥Java>javac HelloWorld.java
HelloWorld.java:4: ';' がありません。
    ]
    ^
エラー 1 個
```

エラーが消えない場合は，以下の項目に注意しながらプログラム全体をもう一度確認すること。

- 全角文字を使ってないか
- 全角スペースを使ってないか
- ( )や{ }等の括弧はきちんと対応がとれているか
- 大文字小文字は間違っていないか

コンパイルに成功すると, 最終的に「C:¥Java¥」ディレクトリの中に「HelloWorld.class」というファイルが作られることになる。これがクラスファイル，もしくはバイトコードと呼ばれ，コンピュータが実行する Java プログラムである。

次にプログラムの実行を行ってみよう。Java プログラムを実行する為には，コマンドプロンプトから java コマンドを利用する。基本的には先程の javac と同じだが，指定する項目がファイル名ではなくプログラム名でなくてはならない。

実際の実行手順は以下のようになる。

```
C:¥Java>java HelloWorld
```

プログラムの 3 行目, “（ダブルクォーテーション） で括られた範囲が画面に表示されれば成功である。

### 3.3.3 Step 2) 売買判定結果, 注文数量, 注文価格を文字列で書く

次に，売買判定結果，注文数量，注文価格を文字列で出力するプログラムを書いてみよう。テキストエディタで以下のように入力する。

```
public class Step2 {
    public static void main(String[] args) {
        System.out.println("Name: " + "Taro");
        System.out.println("ID: " + 1);
        System.out.println("order: " + "SELL");
        System.out.println("price: " + 2000);
        System.out.println("quant: " + 50);
    }
}
```

Java プログラムでは, 原則として main() と記述されている部分から処理が始まり,}（中括弧）の部分まで処理が行われると，プログラムを終了する。中括弧（{ }）でかこまれた部分はブロックと呼ばれている。「main」がついたブロックには，特別に main()メソッドという名前がついている。「メソッド」の意味については後述する。

Java では 1 つの処理の単位を文（statement）と呼び，最後に ;（セミコロン）をつけて記述する。この「文」は，原則として先頭から順番に，1 文ずつ処理されるということになっている。つまり，この例では，プログラムが実行されると main()メソッドの中の 5 つの「文」が上から順番に処理される。

System.out.println は画面に( )内の文字列を出力させるコードであり，文字列を扱う場合は “（ダブルクォーテーション）で括る。+ は二つの文字列を繋げる役割を持った演算子であるので，先程と同様に，上記の例をコンパイルし，実際にプログラムを実行すると以下のように表示される。

```
C:¥Java>java Step2
Name: Taro
ID: 1
order: SELL
price: 2000
quant: 50
```

### 3.3.4 Step 3) ローカル変数の導入

プログラム中で扱うデータを一時的に蓄えておく役割を持ったものを，変数という。メソッドの内部や，さらにその内部のブロックの内部だけで利用されるものを特にローカル変数と呼ぶ。プログラミング言語によって変数の扱いは大きく異なる。具体的には，何も下準備なしでいきなり変数を使うことができる言語と，厳密な準備が必要な言語である。Java は後者の立場をとっており，準備を厳密に行うことで，誤った変数の利用をコンパイルの時点で可能な限り防ぐことができる。

間違った使い方を防ぐための方法の 1 つとして，Java では変数に入るデータの種類をあらかじめ決める必要がある。このデータの種類のことを型という。全ての変数には必ず型を定める必要があり，誤った形式のデータを変数に蓄えようとすると，コンパイル時にエラーとなる。

Java では大きく分けて「基本型」と「参照型」の 2 種類の型がある。基本型では数値と文字の基本的なデータを扱うことができる。それに対し参照型はオブジェクト指向としての特徴，すなわちオブジェクトを扱うための型である。現時点では，基本型とそれ以外の型がある，という程度の理解で十分である。

Java では以下のような基本型があらかじめ 8 種類用意されている。

**基本型･整数型**

| 変数型名 | 最小値 | 最大値 | 入れることができる値 |
|---|---|---|---|
| byte | -127 | 128 | 整数 |
| short | -32768 | 32767 | 整数 |
| int | -2147483648 | 2147483647 | 整数 |
| long | -9.22337E+18 | 9.22337E+18 | 整数 |
| float | 1.4013e-45(精度) | 3.40282E38(精度) | 小数 |
| double | 0(精度) | 1.79769E308(精度) | 小数 |

**基本型･文字型**

| 変数型名 | 入れることができる値 |
|---|---|
| char | ‘で囲まれた文字, unicode |

**基本型･真偽値型**

| 変数型名 | 入れることのできる値 |
|---|---|
| boolean | true,false |

プログラム中では，利用する変数の名前や型を指定してから利用する。これを変数の「宣言」という。Java プログラムでは，全ての変数は必ず宣言を行わなければならない。

宣言した変数にデータを入れることを「代入」という。Java では代入のための記号として「=」が用意されている。「=」の左側に代入を行う変数，右側には代入するデータを書く。

変数の宣言と代入は同時に行うことができる。書式は以下のようになる。

型　変数名　=　値；

型　変数名 1　=　値 1，変数名 2　=　値 2,...；

ローカル変数が使われた例は以下のようになる。

```
public class Step3 {
    public static void main(String[] args) {
        String myName = "Taro";       //String型変数myNameを宣言しTaroを代入
        int myId = 1;                 //int型変数myIdを宣言し1を代入
        String myBuySell = "SELL";    //String型変数myBuySellを宣言しSELLを代入
        int myPrice = 2000;           //int型変数myPriceを宣言し2000を代入
        int myQuant = 50;             //int型変数myQuantを宣言し50を代入

        System.out.println("Name: " + myName);
        System.out.println("ID: " + myId);
        System.out.println("order: " + myBuySell);
        System.out.println("price: " + myPrice);
        System.out.println("quant: " + myQuant);
    }
}
```

このプログラムをコンパイルし，実行すると次のようになる。

```
C:¥Java>java Step3
Name: Taro
ID: 1
order: SELL
price: 2000
quant: 50
```

### 3.3.5 Step 4) フィールドの導入

クラス定義の内部で宣言されるデータを「フィールド(field)」という。メソッドの内部や，さらにその内部のブロックの内部だけで利用されるローカル変数に対し，フィールドはメソッドやコンストラクタのブロックの外に記述され，クラス内のすべてのメソッド，コンストラクタからアクセスすることが可能である。

以下にフィールドを利用した例を示す。

```
public class Step4 {

    private static String myName;
    private static int myId;
    private static String myBuySell;
    private static int myPrice;
    private static int myQuant;
    //クラス内のすべてのメソッド、コンストラクタからアクセスすることが可能

    public static void main(String[] args) {
        myName = "Taro";
        myId = 1;
        myBuySell = "SELL";
        myPrice = 2000;
        myQuant = 50;

        System.out.println("Name: " + myName);
        System.out.println("ID: " + myId);
        System.out.println("order: " + myBuySell);
        System.out.println("price: " + myPrice);
        System.out.println("quant: " + myQuant);
    }
}
```

この例をコンパイルし，実行すると以下のようになる。

```
C:¥Java>java Step4
Name: Taro
ID: 1
order: SELL
price: 2000
quant: 50
```

### 3.3.6 Step 5) メソッドの抽出

プログラム全体としては複雑な処理も，複数の簡単な処理内容を持った部分ごとに分割すれば, 結果的にプログラミングしやすくなることがほとんどである。Java ではこれを「メソッド」と呼んでいる。メソッドの中では，変数の宣言や制御文を利用することができる。

プログラムを部品化して，部品同士がお互いを利用しあう形にすると，プログラム全体の構造を分かりやすくすることができる。メソッドを利用することを，プログラミング用語ではメソッドの呼び出しという。ここでは，式や文（statement）を抜き出し，メソッドとして宣言するとともに，元の式または文（statement）を，抽出されたメソッドに置き換える作業を行う。

メソッドの基本的な記述方法は次のようになる。

```
[修飾子]　戻り値型　メソッド名　（引数の宣言）　{
    //処理内容
}
```

修飾子とは，メソッドについて様々な付加的機能や説明を加えることができる記号である。Java には修飾子がいくつか定義されているが，当面は以下のものを使う。

```
public static
```

戻り値とは，メソッドによって処理された結果得ることができるデータ指す。メソッドを定義する場合，戻り値の型を指定しなければならない。戻り値の型には，int や String などに加えて，メソッドの戻り値専用の型が用意されている。

・void ： 元々は空という意味で，処理結果データが何もないことを表す型

引数はメソッドに処理を行ってもらうために手渡すデータを指す。呼び出される側のメソッドでは，引数はあらかじめ数値が代入された変数として扱うことができる。

以下を参考に実際に作業してみよう。

```
public class Step5 {

    private static String myName;
    private static int myId;
    private static String myBuySell;
    private static int myPrice;
    private static int myQuant;

    public static void setMyName() {
        myName = "Taro";
    }
      //setMyName()メソッドを定義

    public static void setMyId() {
        myId = 1;
    }
      //setMyId()メソッドを定義

    public static void setMyBuySell() {
        myBuySell = "SELL";
    }
      //setMyBuySell()メソッドを定義

    public static void setMyPrice() {
        myPrice = 2000;
    }
      //setMyPrice()メソッドを定義

    public static void setMyQuant() {
        myQuant = 50;
    }
      //setMyQuant()メソッドを定義

    public static void main(String[] args) {
        setMyName();            //setMyName()メソッドの呼び出し
        setMyId();              //setMyId()メソッドの呼び出し
        setMyBuySell();         //setMyBuySell()メソッドの呼び出し
        setMyPrice();           //setMyPrice()メソッドの呼び出し
        setMyQuant();           //setMyQuant()メソッドの呼び出し

        System.out.println("Name: " + myName);
        System.out.println("ID: " + myId);
        System.out.println("order: " + myBuySell);
        System.out.println("price: " + myPrice);
        System.out.println("quant: " + myQuant);
    }
}
```

これをコンパイルし，実行すると以下のように表示される。


### 3.3.7 Step 6) クラスの抽出

これまでクラスは Java プログラムの書き方として暗黙のうちに使ってきたが，ここで正式にクラスの記述方法を紹介する。クラスは，その役割や機能などにより記述方法が変わってくるが，基本となるのは以下の内容である。

```
[修飾子]  class  クラス名  {
    変数宣言
    メソッドの定義
}
```

class はクラス宣言を示す予約語で，続けてクラス名を書く。クラス名の一文字目はアルファベットか__か＄，二文字目以降はこれに加えて数字の利用が可能となる。これは，変数やメソッドにも共通する制約である。なお，Java プログラミングでは一般的にクラス名の頭文字に大文字を使う。

クラスの中には，主に変数の宣言とメソッドを記述することが出来る。クラスの中に宣言した変数は，クラス内部のメソッドから自由に使うことができる。メソッドも同様，クラス内のメソッドから相互に呼び出すことができる。

作成したクラスをプログラム中で利用するためには，new 演算子を利用する。new 演算子は，型枠であるクラスに対する実体をコンピュータのメモリ上に作り出す。これをインスタンスと呼ぶ。New を使うことで，クラスに定義した変数やメソッドを持つインスタンスを作成することができる。

```
New  クラス名（引数１，引数２,...）  ；
```

また，クラスの内部で宣言された変数やメソッドをクラスの外部から利用する場合は，ドットを区切りとして利用する。

変数名.クラス内の変数；

変数名.クラス内のメソッド；

以上のことを踏まえた上で, 各機能を Agent クラスと Main クラスに, 抽出してみよう。ソースは以下の様になる。

Agent クラス

```
public class Agent {

        private String myName;
        private int myId;
        private String myBuySell;
        private int myPrice;
        private int myQuant;

        public String getMyName() {
                return myName;
        }

        public void setMyName(String str) {
                myName = str;
        }

        public int getMyId() {
                return myId;
        }

        public void setMyId(int i) {
                myId = i;
        }

        public String getMyBuySell() {
                return myBuySell;
        }

        public void setMyBuySell(String str) {
                myBuySell = str;
        }

        public int getMyPrice() {
                return myPrice;
        }

        public void setMyPrice(int i) {
                myPrice = i;
        }

        public int getMyQuant() {
                return myQuant;
        }

        public void setMyQuant(int i) {
                myQuant = i;
        }

}
```

Main クラス

```
public class Main {
        public static void main(String[] args) {
                Agent agent = new Agent();
                agent.setMyName("Taro");
                agent.setMyId(1);
                agent.setMyBuySell("SELL");
                agent.setMyPrice(2000);
                agent.setMyQuant(50);

                System.out.println("Name: " + agent.getMyName());
                System.out.println("ID: " + agent.getMyId());
                System.out.println("order: " + agent.getMyBuySell());
                System.out.println("price: " + agent.getMyPrice());
                System.out.println("volume: " + agent.getMyQuant());
    }
}
```

### 3.3.8 Step 7) 継承, getOrder( ) の実装

先述のとおり，Java における継承とは，既に定義されているクラスをもとに，拡張や変更を加えた新しいクラスを定義することである。

具体的な例として, 自動車を表すクラス Car があったとする。次にバスを表すクラス Bus が必要となった場合, 必要なプログラムを最初から全て作るのは時間がかかる。そこで Bus クラスは Car クラスを継承し，Car クラスの持つメソッドや変数を受け継ぐことで，必要最小限のプログラムで Bus クラスを作り上げることができる。

実際のマシン・エージェントのソースを以下に示す。今まで述べてきた概念を使うことにより，getOrder( )の内部が繁雑にならないようにしている。

```
package strategy;

import java.util.*;

public class TestStrategy extends Strategy {

    private Random random;
    private int myBuySell;
    private int myPrice;
    private int myQuant;

    public int getMyBuySell() {
        return myBuySell;
    }

    public void setMyBuySell(int i) {
        myBuySell = i;
    }

    public int getMyPrice() {
        return myPrice;
    }

    public void setMyPrice(int i) {
        myPrice = i;
    }

    public int getMyQuant() {
        return myQuant;
    }

    public void setMyQuant(int i) {
        myQuant = i;
    }

    public TestStrategy(int seed) {
        random = new Random(seed);
    }

    public Order getOrder(int[] spotPrices, int[] futurePrices, int pos,
                          long money, int restDay){
        Order order = new Order();
        setMyBuySell(1);
        setMyPrice(2000);
        setMyQuant(50);
        order.buysell = getMyBuySell();
        order.price = getMyPrice();
        order.quant = getMyQuant();
        return order;
    }
}
```

### 3.4 おわりに

この章では，マシン・エージェント作成に必要となる Java プログラミングについての基礎知識を，実例を交えて紹介した。次章では U-Mart サーバの構築と実験に関して，付随するネットワークの基礎知識を中心に述べていく。

# 第 4 章　サーバ構築と実験

## 4.1 はじめに

ここでは，U-Mart サーバの構築と実際の実験に関して，それに必要なネットワークの基礎知識も含めて説明していく。

## 4.2 ネットワークの基礎知識

### 4.2.1 概要

U‐Mart ネットワーク実験用キットはインターネットを利用して動作する。ここで，インターネットやそれに付随するネットワークの基礎知識を説明しておく。

### 4.2.2 インターネット

インターネットは TCP/IP をベースとするネットワーク・プロトコルによって，世界中のコンピュータを相互接続したネットワークの総称である。ローカルな LAN を相互接続した形態をとっており，インターネットに参加する世界のユーザー同士が相互に通信できるようにしているため，インターネットは良くネットワークのネットワークと呼ばれる。

インターネットの起源は，'69 年に米国国防総省（U.S. Department Of Defense：DOD）の高等研究計画局（Advanced Research Projects Agency：ARPA）が導入した ARPAnet と呼ばれるネットワークである。この ARPAnet では，遠隔地にある複数のコンピュータが接続されていた。中央集中ではなくこのような分散型ネットワークを導入した理由は，一説によると，核攻撃対策を考慮したからだそうだ。つまり中央（たとえばペンタゴン）が攻撃をされても他の施設同士で通信が行えれば指揮系統の乱れが最小限で食い止められるように工夫がされていたわけである。

それから 10 年後の'79 年に，ノースカロライナにある 2 つの大学が，ARPAnet をモデルに相互にネットワーク接続し，BBS サービスや電子メールサービスを開始した。一方'86 年には同じく ARPAnet をモデルとして全米科学財団（National Science Foundation：NSF）が NSFnet と呼ばれるネットワークの運営を開始した。

NSFnet は公共資金が投入されたプロジェクトで，研究を主目的として，大学などの研究機関を中心にネットワークを拡大していった。このように初期のインターネットはネットワークを研究する目的で，大学などの研究機関を中心に発展していた。同様に日本でも，'84 年に大学などの研究機関を中心にして JUNET（Japan Unix NETwork）の運営が開始された。この JUNET では，主に UUCP(Unix to Unix Copy Protocol)というデータ転送方式により，公衆回線を使ったニュースと電子メールのサービスが行なわれ，多くの大学や企業がこれに参加した。

'88 年には専用線による IP 接続を行なう実験ネットワークとして，WIDE（Widely Integrated Distribution Network）が発足した。

'90 年代に入ると冷戦と呼ばれる東西陣営の緊張が緩和をし，ネットワークの軍事利用の需要が低くなった。その頃から，World Wide Web が登場し，現在のように広く商用利用されるようになってきた。

### 4.2.3 プロトコル

#### 4.2.3.1 概要

ネットワークを介してコンピュー同士が通信を行なう上で，相互に決められた約束事の集合。通信手順，通信規約などと呼ばれることもある。

英語しか使えない人と日本語しか使えない人では会話ができないように，対応しているプロトコルが異なると通信することができない。人間同士が意思疎通を行なう場合に，どの言語を使うか(日本語か英語か)，どんな媒体を使って伝達するか(電話か手紙か)，というように 2 つの階層に分けて考えることができるが，コンピュータ通信においても，プロトコルの役割を複数の階層に分けて考える。

階層化することによって，上位のプロトコル(を実装したソフトウェア)は自分のすぐ下のプロトコルの使い方(インターフェース)さえ知っていれば，それより下で何が起きているかをまったく気にすることなく通信を行なうことができる。電話機の操作法さえ知っていれば，NTT の交換局で何が起きているか知らなくても電話が使えるのと同じである。

プロトコルの階層化のモデルは国際標準化機構(ISO)や国際電気通信連合(ITU)などによって 7 階層の OSI 参照モデルとして標準化されており，これに従ってプロトコルを分類することができる。

現在インターネットで標準となっている IP は第 3 層 (ネットワーク層)の, TCP や UDP は第 4 層 (トランスポート層)のプロトコルであり，HTTP や FTP，SMTP，POP などは第 5 層 (セッション層)以上のプロトコルである。

#### 4.2.3.2 ネットワーク/トランスポート層のプロトコル

**TCP/IP**

現在もっとも使われているネットワーク・プロトコルが TCP/IP である。TCP/IP は TCP という送信側と受信側で整合性が約束されたプロトコルと UDP という無手順のプロトコルとその大元になる IP を共有する形になっている。

**IP (Internet Protocol)**

インターネットプロトコル体系の要ともなるネットワーク層プロトコルで RFC791 で規

定されている。ネットワークに参加している機器の住所付け(アドレッシング)や，相互に接続された複数のネットワーク内での通信経路の選定(ルーティング)をするための方法を定義している。

コネクションレス型のプロトコルであるため，確実にデータが届くことを保証するためには，上位層の TCP を併用する必要がある。

UNIX の標準プロトコルとなったことから急速に普及が進み, 現在世界でもっとも普及している。

#### TCP (Transmission Control Protocol)

RFC793 で規定されたインターネットプロトコルで，データの送信/受信側で整合性が約束されたコネクション型プロトコル。ネットワーク層の IP と，セッション層以上のプロトコル(HTTP，FTP，SMTP，POP など)の橋渡しをする。

UDP は転送速度は高いが信頼性が低いが，それに対し TCP は信頼性は高いが転送速度が低いという特徴がある。

#### UDP (User Datagram Protocol)

RFC768 で規定されたインターネットプロトコルで，データの送信/受信側で整合性が約束されないコネクションレス型プロトコル。ネットワーク層の IP と，セッション層以上のプロトコルの橋渡しをする。

#### 4.2.3.3　アプリケーション層のプロトコル

#### HTTP (HyperText Transfer Protocol)

インターネットの World Wide Web サービスにおいて，WWW サーバから WWW クライアントに対して HTML で記述されたドキュメントを送受信するための通信プロトコル。HTTP はリクエストとレスポンスからなる非常に単純なプロトコルで，それぞれリクエストとレスポンスが独立した通信の単位となっている。

#### POP3 (Post Office Protocol Vertion3)

電子メールをスプールしているシステムからスプールの内容を読み出すためのプロトコル。POP は RFC1939 で定義されている電子メール受信のためのプロトコルで，これによりクライアントは，電子メールをスプールしているサーバから，メッセージを取得することが可能になる。この際クライアントとサーバ間でやり取りされるコマンド体系は簡単な ASCII テキスト形式のもので，問い合わせコマンドと応答文字列の対の形式になる。

電子メールの送信に使われる SMTP とセットで利用される。ユーザーがタイトルや発信者を確認する前に，クライアントが全メールを受信してしまうため，発信者やタイトルの一覧を見てから受信するかどうか決められる IMAP を POP の代わりに利用する場合もある。

POP を使うとパスワードがネットワーク上をそのまま流れるため，通信途中で盗まれるかもしれないという危険性がある。この弱点を改善し，パスワードのやり取りを暗号化したものを APOP という。

**SMTP (Simple Mail Transfer Protocol)**

電子メールを送信するためのプロトコル。プロトコルの仕様は RFC821 などで定義されている。SMTP は，もともとはサーバ間でメールをやり取りするためのプロトコルであったが，現在では POP を用いた電子メールクライアント・ソフトが，サーバに対してメールを送信する際にも利用されている。POP3 と同様 SMTP のコマンド体系も簡単な ASCII テキスト形式のもので，問い合わせコマンドと応答文字列の対の形式になる。

**Telnet**

利用するという思想を持って発展してきた。このときユーザーは，端末からホストである UNIX サーバにログインし，他のユーザーと UNIX サーバを共有できるようにする。Telnet は，この端末とホス TCP/IP ネットワークにおいて，リモートにあるサーバを端末から操作できるようにする仮想端末ソフトウェア。またはそれを可能にするプロトコルのこと。歴史的に UNIX は，乏しいコンピュータ資源を複数の端末（ユーザー）で共有トの通信を TCP/IP ネットワーク上で可能にするためのプログラム（プロトコル）である。

#### 4.2.3.4 TCP/IP について

TCP/IP が普及した理由の一つにバークレー版の UNIX でソケットがサポートされたことが大きな原因の一つにある。ソケットとは TCP/IP を使用した接続をプログラム的にひとつのオブジェクトとして扱えるようにした抽象的概念で TCP/IP とアプリケーションの間を仲立ちするインターフェースである。

これにより TCP/IP を継承する上位プロトコルが作られ, そのプロトコルを使用した多くのアプリケーションが作られるようになった。

ソケットは今では Windows 系 OS や MacOS などにも移植され，インターネットアプリケーションが多く作られている。

TCP/IP はそれだけでは抽象的な送受信されるデータを解釈する決まりでしかない。そのため，物理層（電話回線や専用線などのハードウェア）の影響を受けることがない。それを利用することで, 物理的により高速なハードウェアに TCP/IP を載せるといったことも可能となる。今ではインターネット基幹から ATM などで回線を引き出す，といったことも行なわれている。

TCP/IP の中で必ず議論される物の中に IP アドレスがある。IP アドレスとは住所のよう

な物で，つながれているネットワーク内で，必ずユニークなものでなければならない。逆にいうとインターネットにつながれていないネットワークではIP アドレスは勝手にアドレスを割り振っても何の問題もない。

IP アドレスは32bit で成り立っている。32bit というのは16 進数で8 桁の数まで表現できる。16 進数でも 2 進数でも表現してかまわないのだが，通常は分かりやすいように 16 進数を2 桁ずつに分けて10 進数に直しドット区切りで表現する。インターネットに触れた方なら 127.0.0.1 といったような数字の羅列を見かけたことがあるかと思う。これが IP アドレスである。

| 2 進数 | 11000000 | 10101000 | 1 | 10101 |
|---|---|---|---|---|
| 10 進数 | 192 | 168 | 1 | 21 |

現在使われているIP はIPv4 で，先ほど説明したようにIP アドレスを32 ビットで管理している。しかし，近年のインターネットの爆発的な普及により，アドレスが枯渇し始めている。そんな中，IPv6 が策定された。

将来的にはコンピュータだけでなくプリンターやスキャナといった周辺機器や電子レンジ，エアコン，テレビなどの家電から心臓のペースメーカーや医療器具などのネットワーク接続が可能になり，色々なテクノロジーへの転化／転用の可能性があるが，アドレスの絶対数が足りなくなる恐れがある。そのような状況を危惧してIPv6 では32 ビットから64 ビットに拡張されている。これは6 万5 千のネットワーク内の6 万5 千台のコンピュータを認識できていたIP アドレスをさらに約43 億倍に管理できるということである。

次にポートについて説明する。ネットワークにおけるポートとは，データの入り口/出口を指している。IP アドレスがネットワークの住所を表す概念だとすると，ポートはその仕事を表現するための手段である。TCP/IP プロトコルでは，データはソケットごとに管理されるが，接続時に「誰に何のために」その接続を開くのか明示する必要がある。その時の「誰に」がIP アドレスで「何のために」がポートの役割になる。つまりポートにはそれぞれ役割が決められているわけである。

代表的なプロトコルのポートは規定値として以下の番号が割り当てられている。

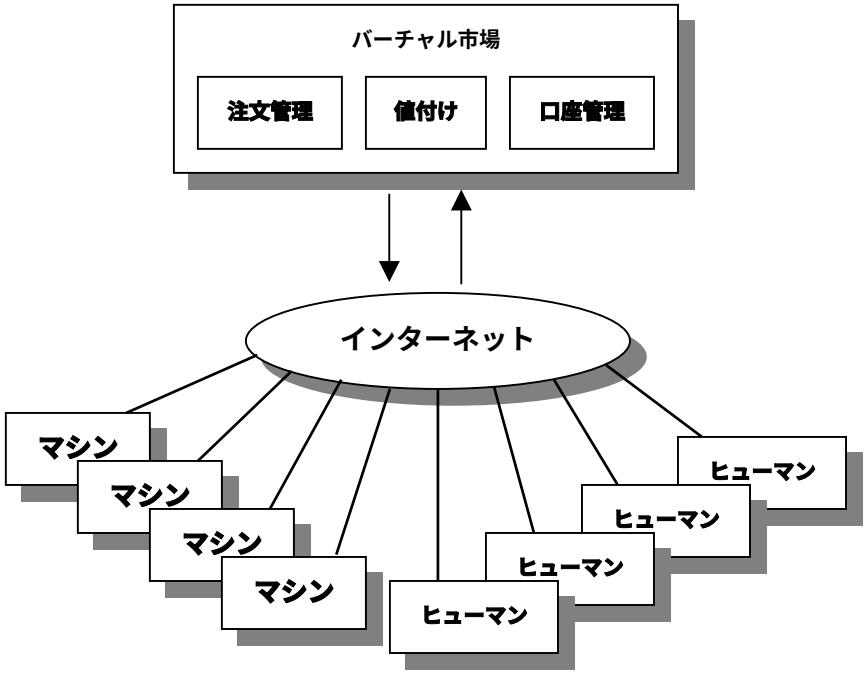
| HTTP | 80 |
|---|---|
| FTP | 21 |
| SMTP | 25 |
| POP3 | 110 |

これらのポート番号は任意に変えることが可能だが，不特定多数とコミュニケーションをとる必要があるネットワークでは規定値として扱われている。また，固有なアプリケー

ションではこのポート番号を他のアプリケーションと，ぶつからないように設計している。

### 4.3 U-Mart サーバの構築と実験

U-Martネットワーク実験用キットのバーチャル市場は技術的には，インターネット上のサーバ・クライアントシステムとして実現する。この概念図を以下に示す。バーチャル市場本体はサーバ上に置かれ，人間の実験参加者やマシン・エージェントなどのクライアントは，そこにインターネットを経由して接続し取引を行なう。また種々の実験の大量・高速な実行を目的としてクライアントをソフトウェア・エージェントだけで構成することも想定している。




実際の実験については，次章の解析編で扱う。

### 4.4 おわりに

U-Mart のネットワーク実験に関連した，ネットワークの基礎知識と，U-Mart システムの概念について述べた。U-Mart を動かすだけならさして重要ではないが，実験を行うに当たって，その背景にある技術について知ることは極めて有意義であると思われる。

# 第 5 章 結論

この論文は，実験編ということで U-Mart プロジェクトの成り立ちから，実際のエージェントの実装，U-Mart サーバの構築と実験へと，U-Mart を利用した学習の下地作りとなるような内容となっている。また，次章のログデータの解析，オブジェクト指向の応用，エージェントの考察からなる解析編への足がかり的な内容となっている。

実際の学習でネックとなると思われる Java プログラミングについても，必要最低限の知識を出来る限り解り易く説明したので，初学者の方々も是非参考にしていただきたい。

今回，卒業論文という形でゼミでの 2 年間の活動を纏めたが，その作業は中々大変なものであった。しかし，その中で何事も前向きにしようという姿勢が重要だということを改めて実感した。私が体験した就職活動もそうだし，普段日常にしている勉学についてもそれは言えることだろう。すぐに結果を求めるのではなく，常にプラス思考で，一生懸命頑張るプロセスこそが大切なのだと思う。

## 謝辞

この論文を作成するにあたり，一緒に協力してくれた原田君，金さんは勿論，この論文の方向性を見出させてくれ，労を惜しまないほど手伝ってくれた石山君には大変感謝している。

また，このゼミでは他には無い貴重な体験をする事ができ，それにより，たくさんの新しい発見をすることができた。その 2 年間の指導教授でもある有賀教授にも感謝したいと思う。先生のおかげで，自分のフィールドが多少なりとも広がったと思う。

## 参考文献

1. 寺野隆雄，プロジェクト：人工市場研究から制度設計へ，進化経済学会大阪千里山大会報告，2002.
2. 石山洸，U-Mart におけるマシン・エージェントの開発，2003.
3. 高橋麻奈，やさしい Java，第 2 版，2002.
4. 三輪賢一，TCP/IP ネットワーク ステップアップラーニング，2003.
5. U-Mart Project Web Server，http://www.u-mart.econ.kyoto-u.ac.jp/
6. 煎り珈琲 Java アプリケーション入門，http://msugai.fc2web.com/java/index.html#objective
7. インターネットの基礎知識，http://www.big.or.jp/~jupiter6/word/inetmenu.html