ソフトウェア第三　講義資料
オブジェクト指向言語 Java

稲葉 雅幸

平成１４年１０月 ３０日（水曜日）

ソフトウェア第二において，Java 言語，Java の仮想計算機の紹介がありました．これから以降のソフトウェア第三では，Java 言語を用いて，さまざまなデータを処理する応用例を紹介してゆきます．

まずは，オブジェクト指向言語の復習からはじめ，リスト処理，グラフィックス処理，並列処理，画像処理，言語処理系，記号処理などを紹介します．

1　Java 言語のプログラミング環境

Java 言語は，\Write Once, Run Anywhere" を哲学とし，どの計算機でも一度書いたプログラムが走るように設計されたプログラミング言語です．

計算機ハードウェア，計算機 OS に依存しないようにするために，仮想的な計算機 (Java virtual machine) を考え，その計算機でプログラムが走り，その仮想計算機がそれぞれの計算機の上で走るという構造を想定しています．

言語自体も，大きなプログラムを作る際には不可欠となっているオブジェクト指向言語となっており，さまざまなクラスライブラリを利用して高度な応用プログラムも容易に作れるようになっています．

Sun という会社が開発し，Java 言語のコンパイラ，仮想計算機が公開されています．

Java ホームページ
http://java.sun.com/

チュートリアル
http://java.sun.com/docs/books/tutorial/index.html

のホームページからダウンロードすることができます．PC でよく使うものは Java Standard Edition(J2SE) で，現在のバージョンは 1.4.1 となっています．

1.1　参考書

最近は，さまざまなテキストが販売されています．全体を展望しやすいように書かれているものとして，

1. Steven Holzner 著，武藤健志 監修，トップスタジオ訳, Java Black Book, インプレス, 2000
2. Michael Morrison , Jerry Ablan(福井真吾，久野禎子，久野靖), Teach Yourself More Java in 21 days(続・Java 言語入門 - 新しいフレームワークとＡＰＩ), ピアソン・エデュケーション,1998
3. 井田昌之, New はやわかり Java,1997
4. Patrick Henry Winston, Sundar Narasimhan(鬼頭繁治，滝沢　徹，牧野祐子訳)，On to Java(ウィンストンの Java), アジソンウェスレイ，1997
5. Bruce Eckel（安藤慶一訳)，Thinking in Java(Bruce Eckel のプログラミングマスターコース, 上下），ピアソン・エデュケーション，1999
6. 大野，前田，井田，松岡，中田，Java プログラミング例題集，bit 別冊，1997
7. Laura Lemay, Charles L. Perkins(武舎広幸，久野禎子，久野靖訳), Teach Yourself Java in 21 days (Java 言語入門 - アプレット，AWT，先進的機能), ピアソン・エデュケーション, 1996

などがあります．

1.2　Java 言語開発環境

java のプログラミング環境である JDK(Java Development Kit) は java.sun.com からダウンロードすることができます．

Windows 上で，走らせるパッケージ JDK(Java development Kit) は，j2sdk-1_4_0-win.exe*　というような実行形式となっていて，これを実行するとコンパイラやクラスライブラリが展開されます．

展開される場所を cygwin の環境ように
c:/cygwin/java/jdk1.4
を指定すると，Cygwin の bash の中では，
/java/jdk1.4
と参照される場所に入ることになります．

実行プログラム探索用パスを指定する環境変数 PATH に，/java/jdk1.4/bin を加えます．

version 1.4 の JDK 環境は以下のようになっています．

```
Htm|Converter.exe*
appletviewer.exe*
extcheck.exe*
idlj.exe*
jar.exe*
jarsigner.exe*
java.exe*
javac.exe*
javadoc.exe*
javah.exe*
javap.exe*
javaw.exe*
jdb.exe*
keytool.exe*
native2ascii.exe*
orbd.exe*
policytool.exe*
rmic.exe*
rmid.exe*
rmiregistry.exe*
serialver.exe*
servertool.exe*
tnameserv.exe*

/java/jdk1.4> ls
COPYRIGHT  bin/ java3d-utils-src.jar  src.zip
LICENSE   demo/     jre/
README.java3d.win32ogl.txt  docs/     lib/
README.txt include/  readme.html

/java/jdk1.4/demo> ls
applets/  jfc/  jpda/        plugin/

/java/jdk1.4/demo> ls -1 applets
Animator/
ArcTest/
BarChart/
Blink/
CardTest/
Clock/
DitherTest/
```

```
DrawTest/
Fractal/
GraphLayout/
GraphicsTest/
ImageMap/
JumpingBox/
MoleculeViewer/
NervousText/
SimpleGraph/
SortDemo/
SpreadSheet/
SymbolTest/
TicTacToe/
WireFrame/
```

という具合に demo プログラムがたくさん用意されています.

1.3 ドキュメント

java のドキュメンテーションも jdk1.4 パッケージと同様にダウンロードし，インストールすることができます．/java/jdk1.4/docs にそのドキュメンテーションがインストールされます．docs ディレクトリには，index.html ファイルが置かれており，それをブラウザで表示すれば，その環境で利用可能なパッケージ，クラス類のドキュメントを調べてゆくことができます.

1.4 コンパイラ javac

コンパイラは javac です．javac は，Java のソースプログラム (通常は拡張子として java をつけたファイル名とする．) を Java 仮想計算機 (JavaVM) の機械語コードへ変換します．ソースプログラムは，通常はクラス名をつけておきます.

コンパイルによって，プログラム内部のクラス名に class という拡張子のついたオブジェクトファイルができあがります.

このプログラムを実行するには，オブジェクトコードファイルを順に実行するプログラムが必要で java という名前のプログラムがそれです.

```
public class HelloWorld {
/**
ここから

ここまでがコメントとなる.
javadoc HelloWorld.java とすると
ドキュメントの html が作られる.
**/
  public static void main(String argv[]){
    System.out.println("Hello World!");
  }
}

コンパイル
% javac HelloWorld.java

実行
% ls *.class
HelloWorld.class

% java HelloWorld
Hello World!
```

この例で，main は HelloWorld のメソッドですが，public がついているのでこのファイルの外からも呼び出すことができるメソッドになっています．また，static がついているので，HelloWorld クラスのクラスメソッドという意味になります．クラスメソッドというのは，クラスのインスタンスが無くても呼び出すことができるメソッドのことです.

System.out.println は，System クラスのクラス変数 out に対してメソッド println を送っているものです.

クラス変数 out の型は，PrintStream クラスのインスタンスになっていて，PrintStream のクラスには，checkError, close, °ush, print, println, write などのメソッドがあります．メソッド println の引数は，引数無，boolean, char, char[], double, °ot, int long, Object, String が可能となっています.

Java では，static がついていないメソッドは，インスタンスメソッドまたはただメソッドと呼びます．これは，インスタンスが作られたあと，そのインスタンスに対して実行するメソッドという意味になります.

また，クラス内の変数に対しても，static がついているものはクラス変数，ついていないものはインスタンス変数といいます．クラス変数は，そのクラスのインスタンスすべてが共有する変数となります.

static 宣言がメソッドについた場合には，インスタンスではなくクラスに対して適用できるメソッドとなり，クラスメソッドまたはスタティックメソッドと呼ばれます．main メソッドは，クラスメソッドになります.

static 宣言が変数についた場合には，その変数はクラス変数になり，つかないものはインスタンス変数となります.

クラスのインスタンスを作った場合に，そのインスタンスからクラス変数もそれ以外の変数（インスタンス変数）も両方がそのインスタンス内部にあるかのように扱うことができますが，クラス変数はそのクラスから作られたすべてのインスタンス間で共通のものとなります．すなわちあるインスタンスでそのクラス変数の値を変更した場合には，他のインスタンスから見るそのクラス変数の値がその変更された値になって見えます．クラス変数は，クラスにひとつ定義され，インスタンス変数は，インスタンスごとに定義されるということになります.

1.5 ドキュメント生成 javadoc

Java 言語環境には，Java のソースプログラムからプログラム内部から，クラスの構造，コメントなどを抜き出して，html 形式のドキュメントファイルを出力するツール javadoc が用意されている.

```
public class HelloWorld {
/**
ここから

ここまでがコメントとなる.
javadoc HelloWorld.java とすると
index.html が作られる.
**/
        public static void main(String argv[]){
                System.out.println("Hello World!");
        }
}

コンパイル
% javadoc HelloWorld.java
```

これにより，

のようなドキュメントが作られます.
public 宣言のないクラスについては，
javadoc -private ¯le.java
のように，-private を指定すればドキュメントが作られます.

1.6 逆アセンブラ javap

javap は，逆アセンブラで，.class ファイルを読み込みます.

```
% javap HelloWorld
Compiled from HelloWorld.java
public class HelloWorld extends java.lang.Object {
    public HelloWorld();
    public static void main(java.lang.String[]);
}
```

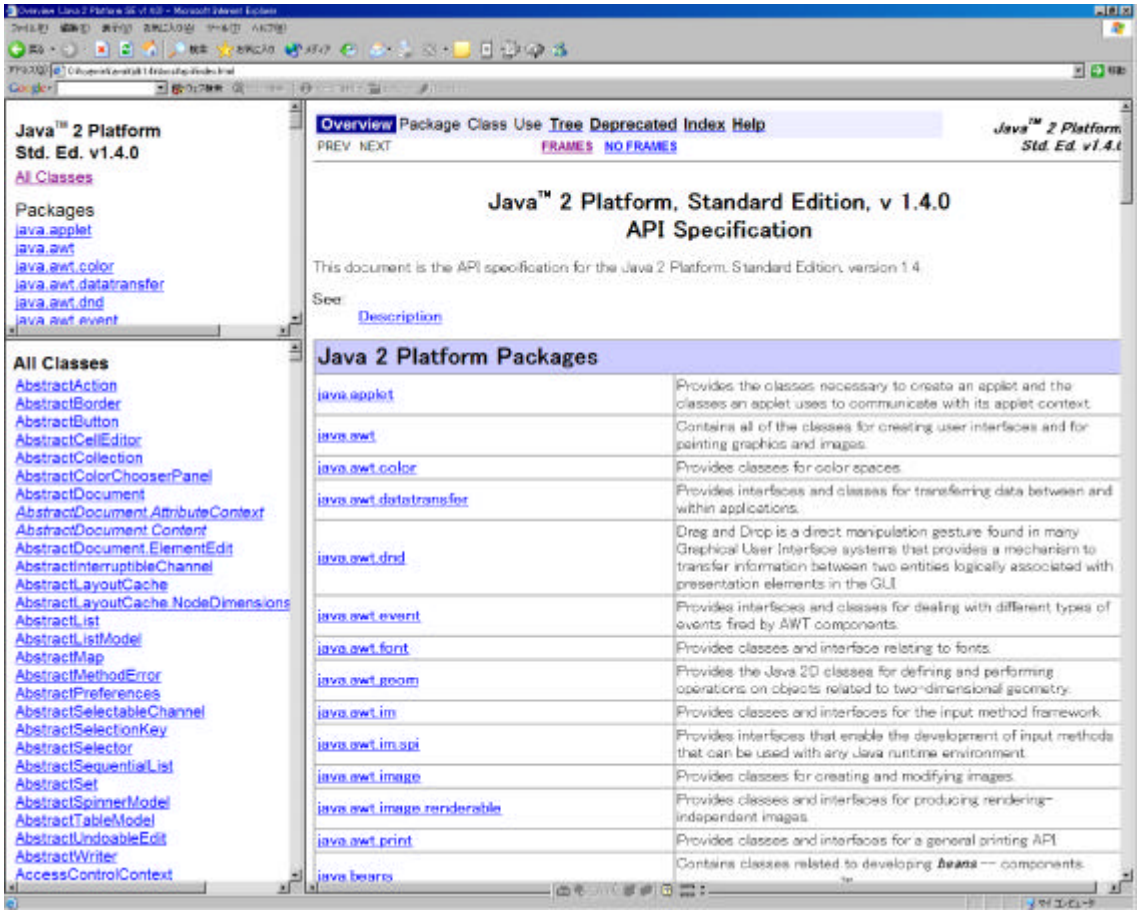

図 1: Java2API Speci¯cation

図 2: HelloWorldDoc

2 オブジェクト指向

オブジェクト指向プログラミングで用いられる概念を定義しておきます.

- **クラス** (class): **同じふるまいをするオブジェクトのグループ.**
- **インスタンス** (instance): **あるクラスの性質をもつひとつのオブジェクト. 具体例. プログラムの実行は, クラスのインスタンスの集合が互いにメッセージをやりとりすることでなされる.**
- **メッセージ** (message): **オブジェクトからオブジェクトへ伝えられる内容.**
- **メソッド** (method): **クラスのメッセージを扱うための手段.**
- **クラス変数** (class variable): **クラス内で宣言され, そのクラスに属するすべてのインスタンスにより共有される変数.**
- **インスタンス変数** (instance variable): **クラス内で宣言され, そのクラスのインスタンスを作ると, そのインスタンスごとに記憶領域が用意される変数.**
- **クラスメソッド** (class method): **インスタンスではなく, クラスに対して適用できるメソッド.**
- **継承** (inheritance): **すでに存在しているクラスに基づいて新しいクラスを定義するための手段. 継承する親のクラスをスーパクラスという. ひとつのスーパクラスのみ継承できるシステムは単一継承といい, 複数の親を継承できるシステムは多重継承という.**
- **委託** (delegation): **あるオブジェクトに対してメッセージが送られてきた場合に, そのオブジェクトの構成要素のひとつに対してメッセージを送ること.**
- **カプセル化** (encapsulation): **オブジェクトの構成要素を外からダイレクトに見えないようにすること.**
- **抽象クラス** (abstract class): **メソッド名などを統一的に表現しておき, サブクラスで実際にそのメソッドの中身を定義してゆくための抽象化されたクラス.**

3 Javaプログラミング

3.1 配列の例

```
// ArrayTest1.java
public class ArrayTest1 {
  public static void main(String argv[]){
    int a1[]={1, 3, 5, 4};
    int a2[]={2, 4, -1};
    int a3[];
    int[] a4;
    int a5[]= new int[5];
    int a6[]= new int[5];
    Integer a7[]= {new Integer(7), new Integer(8)};
    Integer a8[]= new Integer[2];

    a3=a1;
    a6=a2;
    a4=a3;
    a2=a1;
    a8=a7;
//    a8=a2;  -> error

    for (int i=0; i<a1.length; i++)
      System.out.print("a1["+i+"]= "+a1[i]+" ");
    System.out.println();
    for (int i=0; i<a2.length; i++)
      System.out.print("a2["+i+"]= "+a2[i]+" ");
    System.out.println();
    for (int i=0; i<a3.length; i++)
      System.out.print("a3["+i+"]= "+a3[i]+" ");
    System.out.println();
    for (int i=0; i<a4.length; i++)
      System.out.print("a4["+i+"]= "+a4[i]+" ");
    System.out.println();
    for (int i=0; i<a5.length; i++)
      System.out.print("a5["+i+"]= "+a5[i]+" ");
    System.out.println();
    for (int i=0; i<a6.length; i++)
      System.out.print("a6["+i+"]= "+a6[i]+" ");
    System.out.println();
    for (int i=0; i<a7.length; i++)
      System.out.print("a7["+i+"]= "+a7[i]+" ");
    System.out.println();
    for (int i=0; i<a8.length; i++)
      System.out.print("a8["+i+"]= "+a8[i]+" ");
    System.out.println();
  }
}
```

```
> java ArrayTest1
a1[0]= 1 a1[1]= 3 a1[2]= 5 a1[3]= 4
a2[0]= 1 a2[1]= 3 a2[2]= 5 a2[3]= 4
a3[0]= 1 a3[1]= 3 a3[2]= 5 a3[3]= 4
a4[0]= 1 a4[1]= 3 a4[2]= 5 a4[3]= 4
a5[0]= 0 a5[1]= 0 a5[2]= 0 a5[3]= 0 a5[4]= 0
a6[0]= 2 a6[1]= 4 a6[2]= -1
a7[0]= 7 a7[1]= 8
a8[0]= 7 a8[1]= 8
```

3.2 オーバロード

あるクラスに, 引数の数や型は異なるが同じ名前のメソッドを複数定義しておき, 引数の違いによって自動的にメソッドが選ばれて実行される機構のことをいう.

```
// ArrayTest2.java
public class ArrayTest2 {
  static void printArray(String name, int a[]) {
    for (int i=0; i<a.length; i++)
      System.out.print(name + "["+i+"]= "+a[i]+" ");
    System.out.println();
  }
  static void printArray(String name, Object a[]) {
    for (int i=0; i<a.length; i++)
      System.out.print(name + "["+i+"]= "+a[i]+" ");
    System.out.println();
  }
  public static void main(String argv[]){
    int a1[]={1, 3, 5, 4};
    int a2[]={2, 4, -1};
    int a3[];
    int[] a4;
    int a5[]= new int[5];
    int a6[]= new int[5];
    Integer a7[]= {new Integer(7), new Integer(8)};
    Integer a8[]= new Integer[2];

    a3=a1;
    a6=a2;
    a4=a3;
    a2=a1;
  //    a8=a2;  -> error

    printArray("a1",a1);
    printArray("a2",a2);
    printArray("a3",a3);
    printArray("a4",a4);
    printArray("a5",a5);
    printArray("a6",a6);
    printArray("a7",a7);
    printArray("a8",a8);
    printArray("a9",new Object[] {"a", "b", "c"});
    printArray("a10",new Object[]
               {new Float(3.14),
                  new Integer(4),
                  new Double(2.3)});
  }
}
```

```
> java ArrayTest2
a1[0]= 1 a1[1]= 3 a1[2]= 5 a1[3]= 4
a2[0]= 1 a2[1]= 3 a2[2]= 5 a2[3]= 4
a3[0]= 1 a3[1]= 3 a3[2]= 5 a3[3]= 4
a4[0]= 1 a4[1]= 3 a4[2]= 5 a4[3]= 4
a5[0]= 0 a5[1]= 0 a5[2]= 0 a5[3]= 0 a5[4]= 0
a6[0]= 2 a6[1]= 4 a6[2]= -1
```

```
a7[0]= 7 a7[1]= 8
a8[0]= null a8[1]= null
a9[0]= a a9[1]= b a9[2]= c
a10[0]= 3.14 a10[1]= 4 a10[2]= 2.3
```

3.3 例外処理

システムでエラーや例外事象が起きた場合に処理される内容を記述するために try-catch-¯nally 構文があります.

エラーや例外事象をあらわすクラスがあり，それらは Throwable クラスのサブクラスになっています.

```
try {
    try 実行文
} catch (例外型名) {
    catch 実行文1
} catch (例外型名) {
    catch 実行文2
} finally {
    finally 実行文
}
```

```
-----------------------------------------
public void writeList() {
    PrintWriter out = null;

    try {
      System.out.println("Entering try statement");
      out = new PrintWriter(
            new FileWriter("OutFile.txt"));

      for (int i = 0; i < size; i++)
         out.println("Value at: " +
                     i + " = " + victor.elementAt(i));
    } catch (ArrayIndexOutOfBoundsException e) {
        System.err.println(
           "Caught ArrayIndexOutOfBoundsException: " +
           e.getMessage());
    } catch (IOException e) {
        System.err.println(
          "Caught IOException: " + e.getMessage());
    } finally {
        if (out != null) {
            System.out.println("Closing PrintWriter");
            out.close();
        } else {
            System.out.println("PrintWriter not open");
        }
    }
}
```

例外事象を発生させる文としては throw があります.

```
throw someThrowableObject;
```

```
---------------------------
public Object pop() throws EmptyStackException {
    Object obj;

    if (size == 0)
        throw new EmptyStackException();

    obj = objectAt(size - 1);
    setObjectAt(size - 1, null);
    size--;
    return obj;
}
```

4 パッケージ

Java のプログラムライブラリは Package という概念でまとめられています．JDK1.4 のシステムには，次のようなパッケージが用意されています.

```
java.applet
java.awt
java.awt.color
java.awt.datatransfer
java.awt.dnd
java.awt.event
java.awt.font
java.awt.geom
java.awt.im
java.awt.im.spi
java.awt.image
java.awt.image.renderable
java.awt.print
java.beans
java.beans.beancontext
java.io
java.lang
java.lang.ref
java.lang.reflect
java.math
java.net
java.nio
java.nio.channels
java.nio.channels.spi
java.nio.charset
java.nio.charset.spi
java.rmi
java.rmi.activation
java.rmi.dgc
java.rmi.registry
java.rmi.server
java.security
java.security.acl
java.security.cert
java.security.interfaces
java.security.spec
java.sql
java.text
java.util
java.util.jar
java.util.logging
java.util.prefs
java.util.regex
java.util.zip
javax.accessibility
javax.crypto
javax.crypto.interfaces
javax.crypto.spec
javax.imageio
javax.imageio.event
javax.imageio.metadata
javax.imageio.plugins.jpeg
javax.imageio.spi
javax.imageio.stream
javax.naming
javax.naming.directory
javax.naming.event
javax.naming.ldap
javax.naming.spi
javax.net
javax.net.ssl
javax.print
javax.print.attribute
javax.print.attribute.standard
javax.print.event
javax.rmi
javax.rmi.CORBA
javax.security.auth
javax.security.auth.callback
javax.security.auth.kerberos
javax.security.auth.login
javax.security.auth.spi
javax.security.auth.x500
javax.security.cert
javax.sound.midi
javax.sound.midi.spi
javax.sound.sampled
javax.sound.sampled.spi
javax.sql
javax.swing
javax.swing.border
javax.swing.colorchooser
javax.swing.event
javax.swing.filechooser
javax.swing.plaf
javax.swing.plaf.basic
javax.swing.plaf.metal
javax.swing.plaf.multi
javax.swing.table
javax.swing.text
javax.swing.text.html
javax.swing.text.html.parser
javax.swing.text.rtf
javax.swing.tree
javax.swing.undo
javax.transaction
javax.transaction.xa
javax.xml.parsers
javax.xml.transform
javax.xml.transform.dom
javax.xml.transform.sax
javax.xml.transform.stream
org.ietf.jgss
```

```
org.omg.CORBA
org.omg.CORBA_2_3
org.omg.CORBA_2_3.portable
org.omg.CORBA.DynAnyPackage
org.omg.CORBA.ORBPackage
org.omg.CORBA.portable
org.omg.CORBA.TypeCodePackage
org.xml.sax
org.xml.sax.ext
org.xml.sax.helpers
```

4.1 パッケージの利用

パッケージ内のライブラリを使うには import 文を使います．システムの状態を表示するプログラムの例として，

```
// Property.java
import java.util.*;

public class Property {
  public static void main(String[] args) {
    System.out.println(new Date());
    Properties p = System.getProperties();
    p.list(System.out);
    System.out.println("--- Memory Usage:");
    Runtime rt = Runtime.getRuntime();
    System.out.println("Total Memory = "
               + rt.totalMemory()
               + " Free Memory = "
               + rt.freeMemory());
    }
}
```

を実行すると，次のような表示がなされます．

```
% java Property
Wed May 22 03:37:20 JST 2002
-- listing properties --
java.runtime.name=
     Java(TM) 2 Runtime Environment, Stand...
sun.boot.library.path=C:\cygwin\java\jdk1.4\jre\bin
java.vm.version=1.4.0-b92
java.vm.vendor=Sun Microsystems Inc.
java.vendor.url=http://java.sun.com/
path.separator=;
java.vm.name=Java HotSpot(TM) Client VM
file.encoding.pkg=sun.io
user.country=JP
sun.os.patch.level=
java.vm.specification.name=
     Java Virtual Machine Specification
user.dir=e:\home\inaba\lecture\soft402\java\date
java.runtime.version=1.4.0-b92
java.awt.graphicsenv=
     sun.awt.Win32GraphicsEnvironment
java.endorsed.dirs=
     C:\cygwin\java\jdk1.4\jre\lib\endorsed
os.arch=x86
java.io.tmpdir=C:\cygwin\tmp\
line.separator=

java.vm.specification.vendor=Sun Microsystems Inc.
user.variant=
os.name=Windows XP
sun.java2d.fontpath=
java.library.path=
     C:\cygwin\java\jdk1.4\bin;.;C:\WINDOW...
java.specification.name=
     Java Platform API Specification
java.class.version=48.0
java.util.prefs.PreferencesFactory=
     java.util.prefs.WindowsPreferencesFac...
os.version=5.1
user.home=C:\Documents and Settings\inaba
user.timezone=Asia/Tokyo
java.awt.printerjob=sun.awt.windows.WPrinterJob
file.encoding=MS932
java.specification.version=1.4
user.name=INABA
java.class.path=
     .;/java/jdk1.4;/java/jdk1.4/lib;/java...
java.vm.specification.version=1.0
sun.arch.data.model=32
java.home=C:\cygwin\java\jdk1.4\jre
java.specification.vendor=Sun Microsystems Inc.
user.language=ja
awt.toolkit=sun.awt.windows.WToolkit
java.vm.info=mixed mode
java.version=1.4.0
java.ext.dirs=C:\cygwin\java\jdk1.4\jre\lib\ext
sun.boot.class.path=
     C:\cygwin\java\jdk1.4\jre\lib\rt.jar;...
java.vendor=Sun Microsystems Inc.
file.separator=\
java.vendor.url.bug=
     http://java.sun.com/cgi-bin/bugreport...
sun.cpu.endian=little
sun.io.unicode.encoding=UnicodeLittle
sun.cpu.isalist=pentium i486 i386
--- Memory Usage:
Total Memory = 2031616 Free Memory = 1555104
```

自分でパッケージを作る場合は package 文を使います．

4.2 可変サイズのクラス AbstractList

たとえば，java.util パッケージには要素数が可変のデータ構造として，AbstractCollection クラスがあり，その下に，AbstractList と AbstractSet クラスがあります．

AbstractList の方には，Vector,ArrayList,LinkedList などのクラスが用意されています．

```
java.lang.Object
   java.util.AbstractCollection
      java.util.AbstractList
          ArrayList
          Vector
          AbstractSequentialList
              LinkedList
      java.util.AbstractSet
          HashSet
          TreeSet
```

ArrayList,Vector,LinkedList は，AbstractList のサブクラスで，要素数を得るメソッド size と，インデックスを与えて要素を得るメソッド get が AbstractList のメソッドにあります．

```
// ListTest.java
import java.util.AbstractList;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Enumeration;
import java.util.Vector;

public class ListTest {
  static void printVector(String name,
                          Vector a) {
    for (int i=0; i<a.size(); i++)
      System.out.print(" " + name + "["+i+"]= "
                       + a.elementAt(i));
    System.out.println();
  }
  static void printVectorElements(String name,
                                  Vector a) {
    int i=0;
    for (Enumeration e = a.elements();
         e.hasMoreElements(); i++)
      System.out.print(" " + name + "["+i+"]= "
                       + e.nextElement());
    System.out.println();
  }
  static void printList(String name,
                        AbstractList a) {
    System.out.println("Class of " + name +
                       a + " is "+
                       a.getClass().getName());
    for (int i=0; i<a.size(); i++)
      System.out.print(" " + name +
                       "["+i+"]= "+a.get(i));
    System.out.println();
    if (a instanceof Vector) {
      printVector(name,(Vector)a);
      printVectorElements(name,(Vector)a);
    }
  }
  public static void main(String argv[]){

    Vector v1 = new Vector();
    ArrayList v2 = new ArrayList();
    LinkedList v3 = new LinkedList();

    v1.add(new Integer(10));
    v1.add(new Float(15));
    v2.add(new Integer(1));
    v2.add(new Float(2));
    v3.add(new Float(5));
    v3.add(new Double(6));

    printVector("v1",v1);
    printVectorElements("v1",v1);

    printList("v1",v1);
```

```
    printList("v2",v2);
    printList("v3",v3);
    System.out.println();
  }
}
> java ListTest
 v1[0]= 10 v1[1]= 15.0
 v1[0]= 10 v1[1]= 15.0
Class of v1[10, 15.0] is java.util.Vector
 v1[0]= 10 v1[1]= 15.0
 v1[0]= 10 v1[1]= 15.0
 v1[0]= 10 v1[1]= 15.0
Class of v2[1, 2.0] is java.util.ArrayList
 v2[0]= 1 v2[1]= 2.0
Class of v3[5.0, 6.0] is java.util.LinkedList
 v3[0]= 5.0 v3[1]= 6.0
```

Vector クラスの場合は，elementAt メソッドで要素をアクセスしたり，elements メソッドで，Enumeration データを作り，その要素をアクセスする方法などが可能です．

インスタンスからそのクラスを取り出すメソッドとして，getClass があり，その名前を取り出すのに，getName メソッドがあります．また，オブジェクトのクラスを判定するための演算子として instanceof があります．Print-Vector を呼び出すところで，与えられた変数が Vector として宣言されていない場合には，型変換を指示するために，(Vector) という指示を入れています．

4.3 集合クラス AbstractSet

Collection クラスには，AbstractList クラスの他に，AbstractSet クラスがあります．

```
// ListTest1.java
import java.util.AbstractCollection;
import java.util.AbstractList;
import java.util.ArrayList;
import java.util.TreeSet;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.Vector;

public class ListTest1 {
  static Vector v1 = new Vector();
  static ArrayList v2 = new ArrayList();
  static LinkedList v3 = new LinkedList();
  static TreeSet v4 = new TreeSet();
  static TreeSet v5 = new TreeSet();

  static void printList(String name,
                        AbstractCollection a) {
    int i=0;
    for (Iterator e=a.iterator();e.hasNext() ; i++)
      System.out.print(" " + name +
                       "["+i+"]= "+ e.next());
    System.out.println();
  }
  static void prv() {
    printList("v1",v1);
    printList("v2",v2);
    printList("v3",v3);
    printList("v4",v4);
    printList("v5",v5);
    System.out.println();
  }
  public static void main(String argv[]){
    v1.add(new Integer(10));
    v1.add(new Integer(15));
    v2.add(new Integer(10));
    v2.add(new Integer(15));
    v3.add(new Float(10));
    v3.add(new Float(15));
    v4.add(new Float(10));
    v4.add(new Float(15));
    v5.add(new Float(15));
    v5.add(new Float(10));

    if (v1.equals(v1)) System.out.println("v1=v1");
    if (v1.equals(v2)) System.out.println("v1=v2");
    if (v1.equals(v3)) System.out.println("v1=v3");
    if (v2.equals(v3)) System.out.println("v2=v3");
    if (v2.equals(v4)) System.out.println("v2=v4");
    if (v3.equals(v4)) System.out.println("v3=v4");
    if (v4.equals(v5)) System.out.println("v4=v5");

    prv();

  // v1.add(v1);   Stack overflow
    v1.add(v2);
    v3.addAll(v2);
  //    v2.add(v1); Stack overflow
    v4.add(new Float(20));
    v4.addAll(v5);

    prv();
  }
}
> java ListTest1
v1=v1
v1=v2
v4=v5
 v1[0]= 10 v1[1]= 15
 v2[0]= 10 v2[1]= 15
 v3[0]= 10.0 v3[1]= 15.0
 v4[0]= 10.0 v4[1]= 15.0
 v5[0]= 10.0 v5[1]= 15.0

 v1[0]= 10 v1[1]= 15 v1[2]= [10, 15]
 v2[0]= 10 v2[1]= 15
 v3[0]= 10.0 v3[1]= 15.0 v3[2]= 10 v3[3]= 15
 v4[0]= 10.0 v4[1]= 15.0 v4[2]= 20.0
 v5[0]= 10.0 v5[1]= 15.0
```

equals メソッドは，同じデータ型として要素が同じかどうかを調べます．

5 オーバライド

親クラスと子クラスで同じメソッド名のものがあった場合に，子クラスは親クラスのメソッドをオーバライドしているという．子クラスのインスタンスにあるメソッドを作用させた場合に，子クラスに定義されていなければ親クラスのメソッドが実行されるが，子クラスに専用のメソッドを作用させたい場合に子クラスの方にそのメソッドを定義しておく．子クラスのメソッドの中で，親クラスのメソッドも実行したければ明示的に super. メソッド名で呼び出す必要がある．

宇宙空間の浮遊する物体クラス Particle とエンジンをもったロケットクラス Rocket の例を下に示します．

```
// Universe.java
import java.util.*;

public class Universe {
  static int xg = 0, yg = 4;
  static int height = 25, width = 25;

  static class Particle {
    String name;
    int pos[] = {0, 0};
    int dot[] = {0, 0};
    int acc[] = {0, 0};

    Particle(String n, int px, int py,
             int vx, int vy) {
      name = n; pos[0] = px; pos[1] = py;
      dot[0] = vx; dot[1] =vy;
    }
    Particle(String n, int px, int py) {
      this(n,px,py,0,0);
    }
    int[] pos() { return pos; }
    int[] pos(int x, int y) {
      pos[0] = x; pos[1] = y;
      return pos;
    }
    int[] vel() { return dot; }
    public int[] vel(int x, int y) {
      dot[0] = x; dot[1] = y;
      return dot;
    }
    int[] acc() { return acc; }
    int[] acc(int x, int y) {
      acc[0] = x; acc[1] = y;
      return acc;
    }
    void step() {
      pos[0] += dot[0]; pos[1] += dot[1];
      dot[0] += acc[0]; dot[1] += acc[1];
    }
    void status1() {
      System.out.print(name + "'s P:(" +
                       pos[0] + ", " + pos[1]
                       + ")  V:(" +
                       dot[0] + ", " + dot[1]
```

```
                    + ")  A:(" +
                    acc[0] + ", " + acc[1] + ") ");
      if (iscrashed()) {
        System.out.println(";");
        System.out.print("        " + name + " crashed ");
      }
    }
    void status() {
      status1();
      System.out.println();
    }
    boolean iscrashed() {
      return( (pos[0] < width) && (pos[1] > height));
    }
  }

  static class Rocket extends Particle {
    int motor[] ={0, 0};

    public Rocket(String n, int x0, int y0,
                  int vx, int vy, int mx, int my) {
      super(n, x0, y0, vx, vy);
      motor(mx, my);
    }
    int[] motor() { return motor; }
    int[] motor(int x, int y) {
      motor[0] = x; motor[1] = y;
      acc[0] = xg + x; acc[1] = yg + y;
      return motor;
    }
    void status() {
      super.status1();
      System.out.println(" M:(" +
                         motor[0] + ", " +
                         motor[1] + ") ");
    }
  }

  static void main(String argv[]) {
    Particle[] ps = new Particle[6];

    ps[0] = new Particle(new String("p1"),4,3,2,5);
    ps[1] = new Particle("p2",0,0,0,1);
    ps[2] = new Particle("p3",1,1,9,4);
    ps[3] = new Rocket("r1",2,3,4,3,2,1);
    ps[4] = new Rocket("r2",7,5,2,1,7,6);
    ps[5] = new Rocket("r3",2,4,4,2,2,3);

    for (int t=0; t<5; t++) {
      for(int i=0; i<6; i++) {
        ps[i].status();
        ps[i].step();
      }
    }
  }
}
```

実行例は，以下のようになります．

```
> java Universe
p1's P:(4, 3)  V:(2, 5)  A:(0, 0)
p2's P:(0, 0)  V:(0, 1)  A:(0, 0)
p3's P:(1, 1)  V:(9, 4)  A:(0, 0)
r1's P:(2, 3)  V:(4, 3)  A:(2, 5)  M:(2, 1)
r2's P:(7, 5)  V:(2, 1)  A:(7, 10)  M:(7, 6)
r3's P:(2, 4)  V:(4, 2)  A:(2, 7)  M:(2, 3)
p1's P:(6, 8)  V:(2, 5)  A:(0, 0)
p2's P:(0, 1)  V:(0, 1)  A:(0, 0)
p3's P:(10, 5)  V:(9, 4)  A:(0, 0)
r1's P:(6, 6)  V:(6, 8)  A:(2, 5)  M:(2, 1)
r2's P:(9, 6)  V:(9, 11)  A:(7, 10)  M:(7, 6)
r3's P:(6, 6)  V:(6, 9)  A:(2, 7)  M:(2, 3)
p1's P:(8, 13)  V:(2, 5)  A:(0, 0)
p2's P:(0, 2)  V:(0, 1)  A:(0, 0)
p3's P:(19, 9)  V:(9, 4)  A:(0, 0)
r1's P:(12, 14)  V:(8, 13)  A:(2, 5)  M:(2, 1)
r2's P:(18, 17)  V:(16, 21)  A:(7, 10)  M:(7, 6)
r3's P:(12, 15)  V:(8, 16)  A:(2, 7)  M:(2, 3)
p1's P:(10, 18)  V:(2, 5)  A:(0, 0)
p2's P:(0, 3)  V:(0, 1)  A:(0, 0)
p3's P:(28, 13)  V:(9, 4)  A:(0, 0)
r1's P:(20, 27)  V:(10, 18)  A:(2, 5) ;
        r1 crashed  M:(2, 1)
r2's P:(34, 38)  V:(23, 31)  A:(7, 10)  M:(7, 6)
r3's P:(20, 31)  V:(10, 23)  A:(2, 7) ;
        r3 crashed  M:(2, 3)
p1's P:(12, 23)  V:(2, 5)  A:(0, 0)
p2's P:(0, 4)  V:(0, 1)  A:(0, 0)
p3's P:(37, 17)  V:(9, 4)  A:(0, 0)
r1's P:(30, 45)  V:(12, 23)  A:(2, 5)  M:(2, 1)
r2's P:(57, 69)  V:(30, 41)  A:(7, 10)  M:(7, 6)
r3's P:(30, 54)  V:(12, 30)  A:(2, 7)  M:(2, 3)
```

6　マルチスレッド

Java の特徴のひとつに，スレッドが使えるということがあります．

実行のスレッド (thread of execution) とは，並行プログラムにおける逐次実行単位のことである．並行プログラムでは，複数のスレッドが必要なタイミングで互いに同期しつつ，または，排他的に実行を進めてゆきます．

マルチスレッドの機能が必要なものとしては，時々刻々と変化するデータの値をリアルタイムに表示しながら，ユーザが何かの指示をしたい場合に，それを受け付けて，それに対応できないといけない．というときにはマルチスレッド機能が不可欠です．

Java では，スレッドもオブジェクトの一つで，new オペレータで生成されます．生成されただけでは実行がはじまらず，生成されたスレッドに start メソッドを適用するとスレッドは Java システムに内蔵されている実行スケジューラに渡されます．実行スケジューラはスレッド実行の待ち行列にこのスレッドを置いて，待ち行列先頭のスレッドを実行します．

Java ではスレッドは２つの方法で定義できます．

6.1　スレッドクラスを継承したクラスとして

```
// ThreadTest.java
class EchoTread extends Thread {
  int i = 0;
  public void sleep(int a) {
  }
  public void run() {
    while (i++ < 3) {
      System.out.println("hello " + i + " in " + getName());
      try {
        sleep(1000,300);
      } catch (InterruptedException e) {
      }
    }
  }
}
public class ThreadTest {
  public static void main(String argv[]){
    EchoTread th1 = new EchoTread();
    EchoTread th2 = new EchoTread();
    System.out.println("Thread test");
    th1.start();
    th2.start();
    System.out.println("end Start");
  }
}
```

Thread クラスでは，start メソッドが呼ばれると，実行キューに実行すべきスレッドが登録されます．登録されたスレッドが実行状態になるとその Thread の run メソッドが呼ばれるという具合に処理が進みます．

実行してみると以下のようになります．

```
% javac ThreadTest.java
5 java ThreadTest
Thread test
end Start
hello 1 in Thread-1
hello 1 in Thread-2
hello 2 in Thread-1
hello 2 in Thread-2
hello 3 in Thread-1
hello 3 in Thread-2
```

start メソッドが送られてから

```
// Test.java
import java.io.*;
class EchoTread extends Thread {
  int i = 0;
  public void sleep(int a) {
  }
  public void run() {
    while (i++ < 3) {
      System.out.println("hello " + i + " in "
                         + getName());
      try {
        sleep(1000,300);
      } catch (InterruptedException e) {
```

```
    }
  }
 }
}
public class Test {
    public static Runtime run = Runtime.getRuntime();
    public static void main(String argv[]){
        int i=0;
        EchoTread th1 = new EchoTread();
        EchoTread th2 = new EchoTread();
        System.out.println("Thread test");
        th1.start();
        while (i++ < 100) {
            run.gc();
        }
        System.out.println("end Start1");
        th2.start();
        while (i++ < 100) {
            run.gc();
        }
        System.out.println("end Start2");
    }
}
```

という具合に，システムの gc(garbage collection) を呼び出すことによって，時間がかかる処理を挿入すると，次のようになります．

```
% javac Test.java
% java Test
Thread test
hello 1 in Thread-1
hello 2 in Thread-1
end Start1
end Start2
hello 1 in Thread-2
hello 3 in Thread-1
hello 2 in Thread-2
hello 3 in Thread-2
```

ごみ集め

いらなくなったデータを回収して再利用するための処理のことをごみ集め (GC: garbage collection) と呼びます．Java では自動ガーベッジコレクタが用意されており，ヒープ領域に獲得したデータをユーザ自身が明示的に解放処理を実行する必要はありません．複数の Thread が走っている場合には，それぞれの処理の最中にごみが発生しますが，それらを自動的に回収するということが自動的に行われています．

解放処理は，すべてのクラスのルートクラス (object) が最後に行ってほしい finalize メソッドにより実行することも可能です．

6.2 Runnable インタフェースを実現したクラスとして

Java では，クラスは単一継承しかできませんが，多重継承のような仕組みを導入するために，抽象的なクラスとしてメソッドや定数を定義できるインタフェースというものを用意しています．名前はインタフェースですが，インスタンスを作ることができないクラスのようなものです．

Runnable インタフェースは，java.lang パッケージの中に用意されています．ちなみに，java.lang には，CharSequence, Cloneable, Comparable などのインタフェースがあります．

インタフェースの機能を継承する場合には，extends ではなく，implements と書きます．複数のインタフェースを implements することが可能です．

```
// RunnableTest.java
class Echo implements Runnable {
  int i = 0;
  public void run() {
    while (i++ < 3) {
        System.out.println("hello " + i);
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
        }
    }
  }
}
public class RunnableTest {
  public static void main(String argv[]){
      Echo e1 = new Echo();
      Thread th1 = new Thread(e1);
      Echo e2 = new Echo();
      Thread th2 = new Thread(e2);
      System.out.println("Echo test");
      th1.start();
      th2.start();
      System.out.println("end start");
  }
}
```

```
~/lecture/soft402/java/thread> javac RunnableTest.java
~/lecture/soft402/java/thread> java RunnableTest
Echo test
hello 1
end start
hello 1
hello 2
hello 2
hello 3
hello 3
```

実行が開始されていないスレッドが実行待ち行列の先頭になったときに，実行スケジューラは run メソッドを呼び出します．

7 絵を表示する例

絵を表示するサンプルとして，参考書 [井出 97] からのサンプルを以下に示します．

7.1 awt パッケージを用いる場合

Java awt(Abstract window tool) パッケージでは，左上隅を原点とする２次元グラフィック用ライブラリが用意されています．ここには，Frame クラス，Graphics クラスがあります．

このパッケージを使って絵を表示するには，Frame クラスを継承するクラスを作り，new でそのインスタンスを作り，resize で大きさを決め，show で表示します．

```
// Car.java
import java.awt.*;
class Car extends Frame
{
  public boolean handleEvent(Event e)
    {
      if (e.id == Event.WINDOW_DESTROY)
      System.exit(0);
      return super.handleEvent(e);
    }
  public void paint(Graphics g)
    {
      g.clearRect(0,20,400,100);
      g.drawRect(50,60,150,30);
      g.drawOval(80,90,20,20);
      g.drawOval(150,90,20,20);
      g.drawLine(80,60,120,30);
      g.drawLine(120,30,170,30);
      g.drawLine(170,30,190,60);
      g.drawString("Hello",120,80);
    }
  public static void main(String argv[])
    {
      Frame f = new Car();
      f.resize(220,120);
      f.show();
    }
}
```

```
% javac Car.java
% java Car
```

とすると，

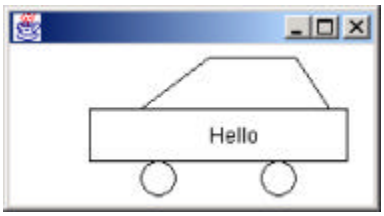

図 3: Javacar

というウィンドウが表示されます．handleevent はウィンドウの右上隅の×点をクリックするとウィンドウが消えてプログラムが終了するためのメソッドです．

7.2 Applet パッケージを用いる場合

Java の applet パッケージを用いると，appletviewer を用いて絵を表示したり，インターネットブラウザを用いて絵を表示することができるようになります．

Applet クラスを継承し，その中に paint メソッドを書いておくと，絵が表示されます．

Applet を継承したクラスのメソッドとして，init(ロード時に実行される処理の記述)，start(開始時に実行される処理の記述)，stop(そのページを離れるときに実行する処理の記述)，paint(実際の描画を行う処理の記述)，destroy(ブラウザを終了させる時に実行する仕事の記述)などを定義するとそれぞれの処理がそのイベントが起こったときに実行される仕組みになっています．

上の Car と同様の絵を表示する場合に，Applet パッケージを利用すると次のようになります．

```
// AppletCar.java
import java.awt.*;
import java.applet.*;
public class AppletCar extends Applet
{
  public void paint(Graphics g)
    {
      g.clearRect(0,20,400,100);
      g.drawRect(50,60,150,30);
      g.drawOval(80,90,20,20);
      g.drawOval(150,90,20,20);
      g.drawLine(80,60,120,30);
      g.drawLine(120,30,170,30);
      g.drawLine(170,30,190,60);
      g.drawString("Hello",120,80);
    }
}
```

これをコンパイルし，次のように，applet として AppletCar.html ファイルの中に記述しておき，appletviewer によって実行すると，

```
% cat AppletCar.html
<APPLET code="AppletCar.class" width=220 height=120>
</APPLET>

実行は,
% appletviewer AppletCar.html
```

となり，次のような絵が表示されます．

8 スレッドを用いたアニメーション

アニメーションは，絵を描く処理を繰り返し実行することによって可能となります．

車の表示を動かすプログラムは以下のようになります．(井出 97 から)

図 4: AppletCar

```
// AnimationCar.java
import java.awt.*;
import java.applet.*;
public class AnimationCar extends Applet
                          implements Runnable
{
  private Thread trig = null;
  private int x;
  public void start() {
    if (trig == null) {
      trig = new Thread(this); trig.start();
    }
  }
  public void stop() {
    if (trig != null && trig.isAlive()) trig.stop();
    trig.stop();
  }
  public void init() {x = 200; }
  public void run() {
    while (trig != null) {
      repaint();
      try { Thread.sleep(125); }
      catch (Exception e) {
           showStatus("Error: " + e);
      }
      x -= 5;
      if (x == 0) x = 200;
    }
  }
  public void paint(Graphics g)
    {
      g.clearRect(0,00,400,100);
      g.drawRect(x+50,40,150,30);
      g.drawOval(x+80,70,20,20);
      g.drawOval(x+150,70,20,20);
      g.drawLine(x+80,40,x+120,10);
      g.drawLine(x+120,10,x+170,10);
      g.drawLine(x+170,10,x+190,40);
      g.drawString("Hello",x+120,60);
    }
  public void update(Graphics g) { paint(g); }
}
```

これを javac でコンパイルし，appletviewer で表示するには，以下のように AnimationCar.html を作る必要があります．

```
% cat AnimationCar.html
<APPLET code="AnimationCar.class" width=220 height=120>
</APPLET>

実行は,
% appletviewer AnimationCar.html
```

ここでの例では，Applet クラスを継承したユーザクラスにさらに，Runnable インタフェースを実装させることでアニメーションを実現しています．

ここでの再表示のためのメソッドとして，repaint は，update を呼び出します．update はその中で画面をクリアーし，paint を呼び出します．画面のクリアーというのは，画面全体をバックグランドカラーで塗りつぶし，フォアグラウンドカラーを設定することです．paint は，実際の描画を行うメソッドです．

9 Java言語仕様

書籍 James Gosling, Bill Joy, Guy Steele, Gilad Bracha: \The Java TM Language Speci¯cation, Second Edition", Addison-wesley, ISBN 0-201-31008-2, 2000. に
http://java.sun.com/docs/books/jls/

Sunの副社長のJames Goslingは，CMUのAndrew Projectや，SunのNeWS()Network-extensible Window System)の開発者です．

Java言語構文のBNF(Backus Naur Form)記述は以下のようになります．

```
[x]{denotes zero or one occurrences of x.
{x} denotes zero or more occurrences of x.
 x | y means one of either x or y.
Identifier:
        IDENTIFIER
QualifiedIdentifier:
        Identifier { . Identifier }
Literal:
        IntegerLiteral
        FloatingPointLiteral
        CharacterLiteral
        StringLiteral
        BooleanLiteral
        NullLiteral
Expression:
        Expression1 [AssignmentOperator Expression1]]
AssignmentOperator:
        =
        +=
        -=
        *=
        /=
        &=
        |=
        ^=
        %=
        <<=
        >>=
        >>>=
Type:
        Identifier { . Identifier } BracketsOpt
        BasicType
StatementExpression:
        Expression
ConstantExpression:
        Expression
Expression1:
        Expression2 [Expression1Rest]
Expression1Rest:
        [ ? Expression : Expression1]
Expression2 :
        Expression3 [Expression2Rest]
Expression2Rest:
        {Infixop Expression3}
        Expression3 instanceof Type
Infixop:
        ||
        &&
        |
        ^
        &
        ==
        !=
        <
        >
        <=
        >=
        <<
        >>
        >>>
        +
        -
        *
        /
        %
Expression3:
        PrefixOp Expression3
        ( Expr | Type ) Expression3
        Primary {Selector} {PostfixOp}
Primary:
        ( Expression )
        this [Arguments]
        super SuperSuffix
        Literal
        new Creator
        Identifier { . Identifier }[ IdentifierSuffix]
        BasicType BracketsOpt .class
        void.class
IdentifierSuffix:
        [ ( ] BracketsOpt . class | Expression ])
        Arguments
        . ( class | this | super Arguments |
              new InnerCreator )
PrefixOp:
        ++
        --
        !
        ~
        +
        -
PostfixOp:
        ++
        --
Selector:
        . Identifier [Arguments]
        . this
        . super SuperSuffix
        . new InnerCreator
        [ Expression ]
SuperSuffix:
        Arguments
        . Identifier [Arguments]
BasicType:
        byte
        short
        char
        int
        long
        float
        double
        boolean
ArgumentsOpt:
        [ Arguments ]
Arguments:
        ( [Expression { , Expression }] )
BracketsOpt:
        {[]}
Creator:
        QualifiedIdentifier
           ( ArrayCreatorRest | ClassCreatorRest )
InnerCreator:
        Identifier ClassCreatorRest
ArrayCreatorRest:
        [ ( ] BracketsOpt ArrayInitializer | Expression ]
            {[ Expression ]}
        BracketsOpt )
ClassCreatorRest:
        Arguments [ClassBody]
ArrayInitializer:
        { [VariableInitializer
            {, VariableInitializer} [,]] }
VariableInitializer:
        ArrayInitializer
        Expression
ParExpression:
        ( Expression )
Block:
        { BlockStatements }
BlockStatements:
        { BlockStatement }
BlockStatement :
        LocalVariableDeclarationStatement
        ClassOrInterfaceDeclaration
        [Identifier :] Statement
LocalVariableDeclarationStatement:
        [final] Type VariableDeclarators ;
Statement:
        Block
        if ParExpression Statement [else Statement]
        for ( ForInitOpt ; [Expression] ; ForUpdateOpt )
              Statement
        while ParExpression Statement
        do Statement while ParExpression ;
        try Block ( Catches | [Catches] finally Block )
        switch ParExpression
           { SwitchBlockStatementGroups }
        synchronized ParExpression Block
        return [Expression] ;
        throw Expression ;
        break [Identifier]
        continue [Identifier]
        ;
        ExpressionStatement
        Identifier : Statement
Catches:
        CatchClause {CatchClause}
CatchClause:
        catch ( FormalParameter ) Block
SwitchBlockStatementGroups:
        { SwitchBlockStatementGroup }
SwitchBlockStatementGroup:
        SwitchLabel BlockStatements
SwitchLabel:
        case ConstantExpression :
        default :
MoreStatementExpressions:
        { , StatementExpression }
ForInit:
        StatementExpression MoreStatementExpressions
        [final] Type VariableDeclarators
ForUpdate:
        StatementExpression MoreStatementExpressions
ModifiersOpt:
        { Modifier }
Modifier:
        public
        protected
        private
        static
        abstract
        final
        native
        synchronized
```

```
        transient
        volatile
        strictfp
VariableDeclarators:
        VariableDeclarator { , VariableDeclarator }
VariableDeclaratorsRest:
        VariableDeclaratorRest { , VariableDeclarator }
ConstantDeclaratorsRest:
        ConstantDeclaratorRest { , ConstantDeclarator }
VariableDeclarator:
        Identifier VariableDeclaratorRest
ConstantDeclarator:
        Identifier ConstantDeclaratorRest
VariableDeclaratorRest:
        BracketsOpt [ = VariableInitializer]
ConstantDeclaratorRest:
        BracketsOpt = VariableInitializer
VariableDeclaratorId:
        Identifier BracketsOpt
CompilationUnit:
        [package QualifiedIdentifier ; ] {ImportDeclaration}
        {TypeDeclaration}
ImportDeclaration:
        import Identifier { . Identifier } [ . * ] ;
TypeDeclaration:
        ClassOrInterfaceDeclaration
        ;
ClassOrInterfaceDeclaration:
        ModifiersOpt
           (ClassDeclaration | InterfaceDeclaration)
ClassDeclaration:
        class Identifier
           [extends Type] [implements TypeList] ClassBody
InterfaceDeclaration:
        interface Identifier
           [extends TypeList] InterfaceBody
TypeList:
        Type { , Type}
ClassBody:
        { {ClassBodyDeclaration} }
InterfaceBody:
        { {InterfaceBodyDeclaration} }
ClassBodyDeclaration:
        ;
        [static] Block
        ModifiersOpt MemberDecl
MemberDecl:
        MethodOrFieldDecl
        void Identifier MethodDeclaratorRest
        Identifier ConstructorDeclaratorRest
        ClassOrInterfaceDeclaration
MethodOrFieldDecl:
        Type Identifier MethodOrFieldRest
MethodOrFieldRest:
        VariableDeclaratorRest
        MethodDeclaratorRest
InterfaceBodyDeclaration:
        ;
        ModifiersOpt InterfaceMemberDecl
InterfaceMemberDecl:
        InterfaceMethodOrFieldDecl
        void Identifier VoidInterfaceMethodDeclaratorRest
        ClassOrInterfaceDeclaration
InterfaceMethodOrFieldDecl:
        Type Identifier InterfaceMethodOrFieldRest
InterfaceMethodOrFieldRest:
        ConstantDeclaratorsRest ;
        InterfaceMethodDeclaratorRest
MethodDeclaratorRest:
        FormalParameters BracketsOpt
           [throws QualifiedIdentifierList] (
        MethodBody | ; )
VoidMethodDeclaratorRest:
        FormalParameters
           [throws QualifiedIdentifierList]
           ( MethodBody | ; )
InterfaceMethodDeclaratorRest:
        FormalParameters BracketsOpt
           [throws QualifiedIdentifierList] ;
VoidInterfaceMethodDeclaratorRest:
        FormalParameters
           [throws QualifiedIdentifierList] ;
ConstructorDeclaratorRest:
        FormalParameters
           [throws QualifiedIdentifierList] MethodBody
QualifiedIdentifierList:
        QualifiedIdentifier { , QualifiedIdentifier}
FormalParameters:
        ( [FormalParameter { , FormalParameter}] )
FormalParameter:
        [final] Type VariableDeclaratorId
MethodBody:
        Block
```

課題

1. **次のプログラムの場合，**aNumber が 3 **であった場合に何を表示するか．プログラムを走らせてみよ．**

```
if (aNumber >= 0)
    if (aNumber == 0)
       System.out.println("first string");
else System.out.println("second string");
System.out.println("third string");
```

2. **プログラムの間違いを指摘して，正しいプログラムにしなさい．**

```
class Rectangle {
    int width;
    int height;
    int area() {
return width*height;
    }
}
public class SomethingIsWrong {
    public static void main(String[] args) {
        Rectangle myRect;
        myRect.width = 40;
        myRect.height = 50;
        System.out.println(
         "myRect's area is " + myRect.area());
    }
}
```

3. **以下のプログラムは何を表示するか．**

```
class StringTest{
    public static void main(String argv[]) {
String s1 = "abc";
String s2 = "abc";
char s3[] = {'a','b','c'};
String s4 = new String(s3);
System.out.println("s1:" + s1 +
   " s2:" + s2 +
   " s3:" + s3 +
   " s4:" + s4);
if (s1 == s2)
    System.out.println("s1 == s2 true");
if (s1 == s4)
    System.out.println("s1 == s4 true");
if (s2 == s4)
    System.out.println("s2 == s4 true");
    }
}
```

4. Universe.java **のシミュレーションをアニメーション表示するプログラムを作れ．締め切り１１月１３日．**