2

2.1

教材を下記のコマンドで実行して、ロボットをゴールまで導いてください。

```
C:¥Java> java -cp Maze.jar;. StartMaze 2
```

解説

だいぶ大きい迷路なので、User.java を気合入れて書いてください。

2.2

以下のプログラムを作成して、コンパイル・実行してみましょう。

```
public class Exercise {
        static public void main(String[] args){
                System.out.println("見事自力でコンパイル完成");
        }
}
```

解説

クラス名が Exercise なので、ファイル名を Exercise.java として保存します。

コンパイルは、

```
C:¥Java> javac Exercise.java
```

実行は、

```
C:¥Java> java Exercise
```

とすれば Ok です。

2.3

以下のプログラムで間違っている箇所を探し、修正してください。

```
public class Sample {

        public static void MAIN(String[] args) {

                System.out.println(おはようございます);
                System.out.println("よろしく"):
```

```
                /*
                System.out.println(こんにちは);
        }
}
```

解説

間違っている箇所は以下のとおりです。

① main が大文字になっている。

② 出力「おはようございます」が”” で囲まれていない。

③ 最後の;(セミコロン)が:(コロン)になっている。

④ コメントが閉じられていない。

```
public class Sample {

        public static void MAIN(String[] args) {

                System.out.println(おはようございます);
                System.out.println(”よろしく”):

                /*
                System.out.println(こんにちは);
        }
}
```

① ② ③ ④

それぞれを修正したプログラムは以下のようになります。

```
public class Sample {

        public static void main(String[] args) {

                System.out.println(”おはようございます”);
                System.out.println(”よろしく”);

                /*
                System.out.println(こんにちは);
                */
        }
```

```
}
```

3 データを保存する～変数～

3.1

以下のデータを格納するのに適した変数を選んでください。

1. 今月のお小遣い（30000 円）
2. 平成 18 年の国家予算(79 兆 6860 億円)
3. 円周率(3.141592...)
4. 文字列「とうきょうとっきょきょきゃきょきゅ」
5. もうすでにご飯を食べたかどうか（yes or no）

解説

1. int(お小遣いが 10 億を超える人は long)
2. long
3. double(ゆとり教育を受けた人は int でも可)
4. String
5. boolean

3.2

以下の計算結果を出力するプログラムを作成してください。

1 1+1

2 10*20

3 半径 10cm の円の面積(ヒント：円の面積はπ×半径の二乗)

4 (10+19)*20

解説

```
public class Exercise2 {
	static public void main(String[] args){
		int x = 1+1;
		System.out.println(x);
```

```
                int y = 10*20;
                System.out.println(y);

                double menseki = 3.14*5*5;
                System.out.println(menseki);

                int z = (10+19)*20;
                System.out.println(z);


        }
}
```

3.3

以下の条件式を boolean 型の変数 lunch に代入するプログラムを作成してください。

1. int 型変数 time がある
2. boolean 型変数 hungry がある。
3. time が 11 以上
4. time が 13 以下
5. hungry が true

ただし、time と hungry の初期値は好きに決めてかまいません。

```
public class Exercise3 {

        public static void main(String[] args) {
                int time=8;
                boolean hungry = true;

                boolean lunch = t  ここに条件式を作成
                System.out.println("お昼にする？:"+lunch);
        }

}
```

回答

```
public class Exercise3 {


	public static void main(String[] args) {
		int time=8;
		boolean hungry = true;


		boolean lunch = time >= 10 && time <= 13 && hungry;
		System.out.println("お昼にする？:"+lunch);
	}


}
```

3.4

1 クラス 40 人の学級で、1 月生まれの人が 5 人いました。1 月生まれの人がこのクラス全体に占める割合(パーセント)を計算するプログラムを作成しました。

次のプログラムが正しく動作するかどうか確認してください。正しく動作しない場合は、正しく動作するように改良してください。

・・・って書いてあるからには、正しく動かないんですけどね。

```
public class Exercise4 {


	public static void main(String[] args) {
		int number = 5;
		int totalNumber = 40;
		int rate = number/totalNumber*100;
		System.out.println("1 月生まれの割合は"+rate+"パーセント");
	}


}
```

解説

int 型の割り算を行う場合は、double にキャストしないといけませんよね。

そこで、次のように修正します。

```
public class Exercise4Answer {
```

```
        public static void main(String[] args) {
                int number = 5;
                int totalNumber = 40;
                double rate = (double)number/totalNumber*100;
                System.out.println("1 月生まれの割合は"+rate+"パーセント");
        }

}
```

4 配列

4.1

配列を作成するには，二つのやり方がありました．

```
String[] weekArray = {“月, “火, “水, “木, “金, “土, “日”};
```

という書き方と，

```
String[] diaryArray = new String[365];
```

という書き方です．

それぞれの特徴を述べてください．

回答

前者の書き方は，初期値が決まっている配列を作成するのに便利です．この場合，一週間の曜日をあらかじめ配列に代入しておくことが出来ます．

一方，後者の書き方はあらかじめデータ数だけ指定して配列を作成しています．中に入れるデータは，後から入れることが可能です．

あらかじめ配列に代入されるべき値が決まっているデータの場合は前者のような書き方を採用し，配列に代入すべき値が決まっていない場合は後者の書き方をします．

この例の場合だと，weekArray は曜日を入れるのであらかじめ値が決まっています．

一方 diaryArray には日記を書き入れるため，配列を作った時点ではどんな文字列が入るかわかりません．今後毎日日記をつけるごとに配列に文字列が代入されていくことでしょう．

4.2

携帯電話の基本使用料金が以下のような携帯電話会社４社があったとします.

この４社の基本料金を一つの配列 basePrice にまとめてください.

A 社：1000 円

B 社：1500 円

C 社：2000 円

D 社：800 円

解説

以下の二通りの方法が考えられます.

```
public class Exercise1 {
	static public void main(String[] args){
		int[] basePrice = {1000,1500,2000,800};
	}
}
```

または,

```
public class Exercise1 {
	static public void main(String[] args){
		int[] basePrice = new int[4];
		basePrice[0] = 1000;
		basePrice[1] = 1500;
		basePrice[2] = 2000;
		basePrice[3] = 800;
	}
}
```

4.3

次のうち正しいものに○，間違っているものには×をつけてください.

1、 配列は変数の一種である

2、 配列は任意の大きさで作成できる.

3、 初期値を決めて配列を作ることが出来る.

4、 配列に入れることの可能なデータ数は保存されないので，自分で管理しておく必要性

がある．

5、配列の大きさ以上の添え字を使って要素を追加しようとしたら，配列の大きさは自動的に変化する．

6、添え字の最小値は 0，最大値は配列の大きさ-1 である．

解答

1、○

2、○

ただし，パソコンのメモリが許す限り．

3、○

4、×

以下のようにすることで，変数 len に配列 array の大きさ 4 が取得できる．

```
int[] array = new int[4];
int len = array.length;
```

5、×

一度作成した配列の大きさは，変化しません．

6、○

4.4

次のような行列①を表現する多重配列を作成して，その行列を画面に表示するプログラムを作成しました．

プログラム中にある空白を埋めてください．

$$\begin{bmatrix} 1 & 5 \\ 2 & 4 \end{bmatrix} \cdots ①$$

```
public class Exercise5 {

    public static void main(String[] args) {
        int[][] matrix = 多重配列の作成

        行列への値の代入
    }
}
```

```
                System.out.println(matrix[0][0]+" "+matrix[0][1]);
                System.out.println(matrix[1][0]+" "+matrix[1][1]);
        }
}
```

解答

```
public class Exercise5 {

        public static void main(String[] args) {
                int[][] matrix = new int[2][2];

                matrix[0][0] = 1;
                matrix[0][1] = 5;
                matrix[1][0] = 2;
                matrix[1][1] = 4;

                System.out.println(matrix[0][0]+" "+matrix[0][1]);
                System.out.println(matrix[1][0]+" "+matrix[1][1]);
        }
}
```

5 制御

5.1

次のプログラムの出力結果を予想してください.
また,「普通だよ」と出力するためにはどこをどう変更すればよいか応えてください.

```
public class Exercise1 {
        public static void main(String[] args) {
                int price = 1000;
                if(price < 500){
                        System.out.println("安いよ");
```

```
                }
                else if(price > 10000){
                        System.out.println("高いよ");
                }
                else{
                        if(price > 2000){
                                System.out.println("高いけど手が届くよ");
                        }
                        else if(price < 1500){
                                System.out.println("お買い得だよ");
                        }
                        else{
                                System.out.println("普通だよ");
                        }
                }
        }
}
```

回答

「お買い得だよ」と表示されます.

「普通だよ」と表示させるには,

```
                int price = 1000;
```

の 1000 を 1500～2000 の間の値に変更します.

5.2

1 から 10 までの合計を求めるプログラムを for 文を使って作成してください.

回答例

これは回答例です. これ以外にも実装方法はあると思います.

```
public class Exercise2 {
        public static void main(String[] args) {
                int sum = 0;
                for(int i = 1; i <= 10; i++){
                        sum += i;
```

```
                }
                System.out.println(sum);
        }
}
```

なお，ポイントは，

```
                for(int i = 1; i <= 10; i++){
                        sum += i;
                }
```

です．

```
                for(int i = 0; i < 10; i++){
                        sum += i;
                }
```

ではうまく計算されませんから注意してください．

5.3

携帯電話の通話料を毎日記録している配列 price があります．この配列を調べて，何日目で通話料が 1000 円を越えたかを調べるプログラムを作成しました．

空白部分を生めて完成させてください．

```
public class Exercise3 {

        public static void main(String[] args) {
                int[] price = {10,50,150,32,485,21,150,100,456,578,400};

                int total = 0; //通話料合計
                int i = 0; //日にち
                while(
                        i++;
                }
                System.out.println(i+"日目に 1000 円を超えました");

        }

}
```

While 文の条件

通話料合計の計算

回答例

```
public class Exercise3 {


	public static void main(String[] args) {
		int[] price = {10,50,150,32,485,21,150,100,456,578,400};


		int total = 0;
		int i = 0;
		while(total < 1000){
			total += price[i];
			i++;
		}
		System.out.println(i+"日目に 1000 円を超えました");


	}


}
```

5.4

以下の二つの行列がある．この行列を足し合わせた結果を表示してください．

$$\begin{bmatrix} 1 & 5 \\ 2 & 4 \end{bmatrix} \cdots ①$$

$$\begin{bmatrix} -1 & 2 \\ 2 & 1 \end{bmatrix} \cdots ②$$

```
public class Exercise4 {


	public static void main(String[] args) {
		int[][] matrix1 = {
					{1, 5},
					{2, 4}
		};
```

```
                int[][] matrix2 = {
                                {-1, 2},
                                {2, 1}
                };

                int[][] matrix3 = new int[2][2];

        ここで matrix1 と matrix2 を足し合わせる計算をする

                System.out.println(matrix3[0][0]+" "+matrix3[0][1]);
                System.out.println(matrix3[1][0]+" "+matrix3[1][1]);
        }

}
```

回答例

```
public class Exercise4 {

        public static void main(String[] args) {
                int[][] matrix1 = {
                                {1, 5},
                                {2, 4}
                };

                int[][] matrix2 = {
                                {-1, 2},
                                {2, 1}
                };

                int[][] matrix3 = new int[2][2];

                for(int i = 0; i < 2; i++){
                        for(int j = 0; j < 2; j++){
                                matrix3[i][j] = matrix1[i][j]+matrix2[i][j];
                        }
                }
```

```
                System.out.println(matrix3[0][0]+" "+matrix3[0][1]);
                System.out.println(matrix3[1][0]+" "+matrix3[1][1]);
        }

}
```

ポイントは for 文を二回使うところです．また，for 文の中に for 文を使う場合，同じ添え字は使えませんので注意しましょう．ここでは，最初の for 文では i をつかって，次の for 文では j を使っています．

5.5

第 2 章の迷路プログラムの User.java を書き換えて，壁に当たるまで右に進み，壁にあたったらまた壁に当たるまで下に進むロボットを作成してください．

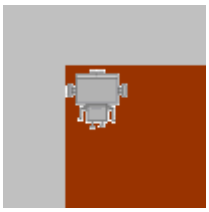
なお，右側が壁かどうかを確認するためには，

```
boolean right = robot.isRightWall();
```

によって確認できます．ロボットの右側が壁だった場合，right に true が代入されます．同様に，

```
boolean left = robot.isLeftWall(); //左側が壁かどうか
boolean up = robot.isUpWall();//上側が壁かどうか
boolean down = robot.isDownWall();//下側が壁かどうか
```

で，上下左右全ての方向が壁かどうかを確認することが出来ます．



このような状態だと，

right:false

left:true

up:true

down:false

になります．

回答例

User.java を以下のようにします.

```
import jp.co.shuwasystem.java.maze.map.RobotManager;
import jp.co.shuwasystem.java.maze.map.Robot;



public class Sample implements RobotManager{

	public void action(Robot robot) {
		while(!robot.isRightWall()){
			robot.right();
		}
		while(!robot.isDownWall()){
			robot.down();
		}
	}
}
```

まず，while 文を使って右側が壁でない限り右に進ませます．次に，同様に while 文を使って下が壁でない限り下に進みます.

このプログラムを実行すれば，右から下へ移動するので,

```
C:¥Java>java -cp Maze.jar;. StartMaze
```

によって実行した迷路ではないフィールド上ならば，ロボット君はゴールまで自動的にゴールまでたどり着いてくれるはずです.

6

6.1

整数（int）型の引数を取り，返値が実数(double)型であるメソッド(method)の書き方として正しいものを選んでください.

1，

```
static void method(int a, double j){
	//処理
}
```

2,

```
static double method(int a){
```

```
        //処理
}
```

3,

```
static int method(double a){
        //処理
}
```

回答

2

解説

1 は，引数に int と double を引数に取るメソッドです.

3 は，double 型を引数にとって，int 型を返値とするメソッドです.

6.2

以下の記述のうち正しいものに○を，間違っているものに×をつけてください.

1 メソッドの引数は 2 つ以上作ることが出来る

2 メソッドの返値は 2 つ以上作ることが出来る

3 メソッドの引数は必ず必要である.

4 メソッドの返値は必ず必要である.

5 メソッドの引数はすべて同じ型でなければいけない

回答

○，×，×，×，×

6.3

以下のプログラムは，ある金額を入れるとそれに応じた税込み価格を計算してくれるメソッドです.

このメソッドが正しく動くかどうか検証し，動かない場合は正しく動くようにプログラムを修正してください.

```
public class Exercise3 {
        static public void main(String[] args){
                double price = 100;
                taxPrice(price);
```

```
                System.out.println("税込み価格は"+price);
        }

        static void taxPrice(double price){
                price = price * 1.05;
        }
}
```

回答

taxPrice メソッドの中で price の値を変更していますが, これでは main メソッドの price の値は変化しません.

以下のように変更するのが一つの方法です.

```
public class Exercise3Answer {
        static public void main(String[] args){
                double price = 100;
                price = taxPrice(price);
                System.out.println("税込み価格は"+price);


        }

        static double taxPrice(double price){
                price = price * 1.05;
                return price;
        }

}
```

これ以外にも方法はあると思いますので，考えてみてください.

6.4

実数(double)型の配列を引数として，その平均値を求めるメソッドを作成してください.

回答例

実数型の配列には，2006 年 FIFA ワールドカップに出場した日本代表チームの身長を代入して，その平均身長を求めてみました.

```
public class Exercise4 {
	public static void main(String[] args) {
		double[] heightAry = {185,184,179,181,171,176,182,178,179,177,187,178,175,173,178,181,181,175,180,184,177,177,173};
		double avg = average(heightAry);
		System.out.println("2006 年 FIFA ワールドカップ日本代表の平均身長は");
		System.out.println(avg+"cm");
	}

	static double average(double[] ary){
		double average = 0;
		for(int i = 0; i < ary.length; i++){
			average += ary[i];
		}
		average /= ary.length;
		return average;
	}
}
```

7 JAVA の本質に触れる ～クラス～

7.1

次の記述のうち正しいものに○を，間違っているものに×をつけてください.

1、クラスを記述するファイルの名前は好きに決めていい

2、クラスには必ずコンストラクタを作らなければいけない

3、フィールドとは一種の変数のことである.

4、クラスの変数は，インスタンスを作らなければ利用できない

5、クラス名と同じ名前のメソッドは作れない

解答

1、×

必ず，クラス名.java という名前にしなければいけません．クラス名が PersonalData ならば，PersonalData.java という名前のファイルにクラスの内容を記述します.

2、×

コンストラクタは必ずしも作る必要はありません.

3、○

4、○

単に変数を宣言しただけでは利用できません.

5、○

クラス名と同じ名前のメソッドは，自動的にコンストラクタとして扱われます.

7.2

以下は，PersonalData を使って住所録を作成しようとするプログラムです.

間違っている箇所を見つけて，修正してください.

```
public class Exercise1 {


	public static void main(String[] args) {
		PersonalData personalData;
		personalData.name = "名前";
		personalData.address = "住所";
		personalData.phoneNumber = "電話番号";
		personalData.emailAddress = "E-Mail アドレス";
	}


}
```

回答

PersonalData 型の変数を確保していますが，インスタンスを作成していません．インスタンスを作成するため，以下のように変更します.

```
public class Exercise1 {


	public static void main(String[] args) {
		PersonalData personalData = new PersonalData();
		personalData.name = "名前";
		personalData.address = "住所";
		personalData.phoneNumber = "電話番号";
		personalData.emailAddress = "E-Mail アドレス";
	}
```

```
}
```

7.3

以下のようなクラスがあったとき,①～⑩までが下記の A～B のどれに当てはまるか応えてください.

A:フィールド
B:メソッド
C:コンストラクタ

```
public class Exercise2 {

        int score;     //①
        int number; //②
        String answer; //③

        public Exercise2(){                    ┐
                score = 10;                    │
                number = 2;                    ├ ④
        }                                      ┘

        public Exercise2(int s){               ┐
                score = s;                     │
                number = 2;                    ├ ⑤
        }                                      ┘

        public int getScore(){                 ┐
                return score;                  ├ ⑥
        }                                      ┘

}
```

回答

① A
② A
③ A
④ C
⑤ C
⑥ B

7.4

以下のような機能を持つクラスを作成してください.

1、 double 型の配列をフィールド
2、 引数が実数(double)型の配列であるコンストラクタ
3、 配列の要素の平均を求めるメソッド
4、 配列の要素の合計を求めるメソッド
5、 配列の要素の最大値を求めるメソッド
6、 配列の要素の最小値を求めるメソッド
7、 指定した値が配列内に存在するかどうかを調べるメソッド

クラス実装例

```
public class Exercise4 {

        double[] ary;


        /**
         * 実数型の配列を引数にしたコンストラクタ
         */
        public Exercise4(double[] a) {
                コンストラクタ
        }


        /**
         * 要素の合計を求めるメソッド
         */
        public double sum(){
                要素の合計を求める
        }
```

```
    /**
     * 要素の平均を求めるメソッド
     */
    public double average(){
        要素の平均を求める
    }

    /**
     * 要素の最大値を求めるメソッド
     */
    public double max(){
        要素の最大値を求める
    }

    /**
     * 要素の最小値を求めるメソッド
     */
    public double min(){
        要素の最小値を求める
    }

    /**
     * 値を要素に持っているかどうかを確認するメソッド
     */
    public boolean hasElement(double d){
        等しい要素を持っているかどうか調べる
    }

}
```

回答例

```
public class Exercise4 {
	double[] ary;

	/**
	 * 実数型の配列を引数にしたコンストラクタ
	 */
	public Exercise4(double[] a) {
		ary = a;
	}

	/**
	 * 要素の合計を求めるメソッド
	 */
	public double sum(){
		double sum = 0;
		for(int i = 0; i < ary.length; i++){
			sum += ary[i];
		}
		return sum;
	}


	/**
	 * 要素の平均を求めるメソッド
	 */
	public double average(){
		double sum = 0;
		for(int i = 0; i < ary.length; i++){
			sum += ary[i];
		}
		return sum/ary.length;
	}

	/**
	 * 要素の最大値を求めるメソッド
	 */
```

```
	public double max(){
		double max = -999999;
		for(int i = 0; i < ary.length; i++){
			if(max < ary[i]){
				max = ary[i];
			}
		}
		return max;
	}

	/**
	 * 要素の最小値を求めるメソッド
	 */
	public double min(){
		double min = 999999;
		for(int i = 0; i < ary.length; i++){
			if(min > ary[i]){
				min = ary[i];
			}
		}
		return min;
	}

	/**
	 * 値を要素に持っているかどうかを確認するメソッド
	 */
	public boolean hasElement(double d){
		for(int i = 0; i < ary.length; i++){
			if(ary[i] == d){
				return true;
			}
		}
		return false;
	}

}
```

なお，最小値が 999999 以上だった場合と，最大値が-999999 以下だった場合は，最大値最小値を正しく求めることは出来ません.

そういった値を含む配列で正しく動作させるには工夫が必要です.

ちなみに，利用例はこんな感じ.

```
	public static void main(String[] args) {
		double[] heightAry = {185,184,179,181,171,176,182,178,179,177,187,178,175,173,178,181};

		Exercise4 ex = new Exercise4(heightAry);
		System.out.println("合計値"+ex.sum());
		System.out.println("平均値"+ex.average());
		System.out.println("最大値"+ex.max());
		System.out.println("最小値"+ex.min());
		System.out.println("176 があるかどうか:"+ex.hasElement(176));
	}
```

8 クラスの機能を追加する～継承とインターフェース

8.1

下記のクラスについて正しい記述を選んでください.

1、 RobotManager クラスは User クラスを継承している

2、 User クラスは RobotManager クラスを継承している

3、 RobotManager クラスは User インターフェースを実装している

4、 User クラスは RobotManager インターフェースを実装している

```
import jp.co.shuwasystem.java.maze.map.RobotManager;
import jp.co.shuwasystem.java.maze.map.Robot;



public class Sample implements RobotManager{
	public void action(Robot robot) {
```

```
                robot.right();
        }
}
```

解答

4

implements は継承を表すキーワードなので，User クラスは RobotManager インターフェースを実装していることになります.

8.2

以下のうち，正しい説明に○を，間違っている説明に×を付けてください.

1、 どんなクラスからでも派生クラスを作ることが出来る

2、 一つのクラスから複数のクラスを派生させることが出来る

3、 複数のクラスを継承したクラスを作ることが出来る

4、 複数のインターフェースを実装することが出来る

5、 abstract メソッドを持つクラスは abstract クラスになる

6、 インターフェースにはメソッドの定義しか書くことが出来ない

解答

1、 ×

final 修飾子のついたクラスを継承することは出来ません

2、 ○

3、 ×

4、 ○

5、 ○

6、 ×

定数の定義を書くことが出来ます.

8.3

第 2 章で使った User.java クラスの action メソッドを消してしまうと，コンパイルすることができません.

その理由を答えてください.

```
import jp.co.shuwasystem.java.maze.map.RobotManager;
```

```
import jp.co.shuwasystem.java.maze.map.Robot;



public class Sample implements RobotManager{
	/*
	削除
	protected void action(Robot robot) {
	}
	*/
}
```

```
User.java:5: User は abstract でなく、jp.co.shuwasystem.java.maze.map.RobotManag
er 内の abstract メソッド action(jp.co.shuwasystem.java.maze.map.Robot) をオー
バーライドしません。
public class User implements RobotManager {
       ^
エラー 1 個
```

解答

RobotManager はインターフェースで, action メソッドが定義されています. そのため, 実装したクラス User で action メソッドを定義しないといけません.

8.4

下記のインターフェース Figure は面積を求めるメソッドを持っています.

このインターフェース Figure を実装した三角形クラス Triangle を作成しました. 同様に, 正方形クラス Square, 長方形クラス Rectanble を作成してみてください.

```
public interface Figure {
	double getArea();
}
```

```
public class Triangle implements Figure {
```

```
	double width;
	double height;

	public Triangle(double w, double h){
		width = w;
		height = h;
	}

	public double getArea() {
		return width*height/2;
	}

}
```

実行例

```
public class Exercise4 {
	static public void main(String[] args){
		Triangle triangle = new Triangle(10, 10);
		System.out.println("底辺 10cm, 高さ 10cm の三角形の面積は"+triangle.getArea());
	}
}
```

```
底辺 10cm, 高さ 10cm の三角形の面積は 50.0
```

解答例

```
public class Square implements Figure {

	double width;

	public Square(int w) {
		width = w;
	}

	public double getArea() {
		return width*width;
```

```
	}


}
```

```
public class Rectangle implements Figure {


	double width;
	double height;


	public Rectangle(int w, int h) {
		width = w;
		height = h;
	}


	public double getArea() {
		return width*height;
	}



}
```

利用例：

```
public class Exercise4 {
	static public void main(String[] args){
		Triangle triangle = new Triangle(10, 10);
		System.out.println("底辺 10cm, 高さ 10cm の三角形の面積は"+triangle.getArea()+"平方
cm");


		Square square = new Square(10);
		System.out.println("一辺 10cm の正方形の面積は"+square.getArea()+"平方 cm");


		Rectangle rectangle = new Rectangle(5, 5);
		System.out.println("横 5cm, 縦 5cm の長方形の面積は"+rectangle.getArea()+"平方 cm");
```

```
        }
}
```

実行結果：

```
底辺 10cm，高さ 10cm の三角形の面積は 50.0 平方 cm
一辺 10cm の正方形の面積は 100.0 平方 cm
横 5cm，縦 5cm の長方形の面積は 25.0 平方 cm
```

9 パッケージと修飾子

9.1

パッケージに関する以下の記述のうち正しいものに○，間違っているものに×をつけてください.

1、パッケージは必ず書く必要がある

2、パッケージはクラス宣言の前ならば任意の場所に記述できる.

3、クラスは任意のパッケージにおくことが出来る

4、インポートせずに利用できるクラスは同一パッケージにあるクラスのみである

解答

1、×

パッケージを書かないとデフォルトパッケージに所属することになります.

2、×

インポート宣言，及びクラス宣言の前に書く必要があります．通常は.java ファイルの先頭に書きます.

3、×

ディレクトリと同期したパッケージ名を記述します.

```
Java のプログラムを書いているディレクトリ¥hoge¥piyo¥
```

にある java ファイルは

```
package hoge.piyo;
```

に属することになります.

4、×

同一パッケージのクラスもインポートせずに使えますが，それ以外にも，java.lang パッケージにあるクラスはインポートせずに利用することが出来ます.

9.2

以下のクラスが各パッケージに所属しています.

1、 sports

```
class Ski{
}
```

```
class Chess{
}
```

2、 sports.ball

```
class BaseBall{
}
```

```
class Soccor{
}
```

このとき，以下の記述をしたときに利用可能となるクラスを全て列挙してください.

1、

```
import sports.Ski
```

2、

```
import sports.*;
```

3、

```
import sports.ball.Soccor;
```

4、

```
import sports.*;
import sports.ball.*;
```

解答

1、 Ski

2、 Ski, Chess

下位パッケージに含まれる BaseBall, Soccor は含まれないことに注意

3、 Soccor

4、 Ski, Chess, BaseBall, Soccor

9.3

下記のクラスにあるフィールドを private にした上で，外部から値を設定したり，取得したりできるように変更してください.

ただし，hour は 0-23 の間，minute は 0-59 の間，second は 0-59 の間の値しか取れないように制限を加えてください.

```
public class Clock {

	int hour;
	int minute;
	int second;

}
```

解答

```
public class Clock {

	private int hour;
	private int minute;
	private int second;

	public int getHour() {
		return hour;
	}

	public void setHour(int h){
		if(h < 24 && h >= 0){
			this.hour = h;
		}
	}

	public int getMinute() {
```

```
                return minute;
        }

        public void setMinute(int m) {
                if(m < 60 && m >= 0){
                        this.minute = m;
                }
        }

        public int getSecond() {
                return second;
        }

        public void setSecond(int s) {
                if(s < 60 && s >= 0){
                        this.second = s;
                }
        }
}
```

9.4

問題 9.2 で作成した Clock クラスに，この時計の値段をあらわす整数(int) 型のフィールド price を追加してください.

ただし，以下の条件を満たすように設計してください.

1、 price フィールドは全ての Clock インスタンスで必ず同一の値になる

2、 price フィールドを利用できるのは Clock クラスからのみ.

3、 price フィールドの値を取得するメソッドはどのクラスからでも利用可能

4、 price フィールドの値を設定するメソッドは同一パッケージからしかできない

5、 price フィールドの初期値は 1000

解答

下記の記述を Clock クラスに追加します.

```
public class Clock {
```

```
        //中略

        private static int price = 1000;

        public int getPrice(){
                return price;
        }

        void setPrice(int p){
                price = p;
        }
}
```

getPrice と setPrice メソッドは，static 修飾子を付けても Ok です.

10　便利なクラス JavaAPI

10.1

以下の説明のうち正しいものに○，間違っているものに×をつけてください.

1、 JavaAPI はすべてインポートしなくても使うことが出来る

2、 すべてのプリミティブ型には対応するラッパークラスが存在する

3、 配列で可能なことはすべて ArrayList でも可能である

4、 HashSet には重複するデータがないことが保障されている

5、 HashMap で存在しないキーを使ってデータを取得しようとするとプログラムが停止する

解答

1、 ×

ArrayList などは java.util パッケージにあるためインポートしないと使えません. String など java.lang にあるクラスのみインポートせずに使えます.

2、○

3、×

例えば，要素数を指定して作成することは ArrayList では出来ません.

4、○

5、×

単に null が返されます.

10.2

URL が文字列として与えられたとき，その URL をサーバ名とファイル名に分割してください.

例えば,

http://www.example.co.jp/expdir/target/filename.html

という URL があった場合，サーバ名は，www.example.co.jp，ファイル名は /expdir/target/filename.html となります.

なお，URL は以下のような条件にしたがって作成されているとします.

1、http://から始まる

2、サーバ名とファイル名の区切りは「/」

3、ファイル名はない可能性がある（http://www.example.co.jp のように）

```
public class Exercise1 {

        public static void main(String[] args) {

                String url = "http://www.example.co.jp/expdir/target/filename.html";
                String server = "";
                String file = "";

                ここで server にサーバ名を，file にファイル名を代入します

                System.out.println("サーバ名:"+server);
                System.out.println("ファイル名:"+file);

        }
```

```
}
```

解答例

```
public class Exercise1 {

	public static void main(String[] args) {

		String url = "http://www.example.co.jp/expdir/target/filename.html";
		String server = "";
		String file = "";

		url = url.replaceFirst("http://", "");
		String[] urlAry = url.split("/");
		server = urlAry[0];
		if(urlAry.length > 1){
			for(int i = 1; i < urlAry.length; i++){
				file += "/"+urlAry[i];
			}
		}

		System.out.println("サーバ名:"+server);
		System.out.println("ファイル名:"+file);

	}

}
```

10.3

7.4 で作成したクラスを ArrayList を使って作り直してみてください.
なお，このクラスは以下のような機能を持っているものとします.

1、引数無しのコンストラクタ
2、新しく要素（Double 型）を追加するメソッド

3、配列の要素の平均を求めるメソッド
4、配列の要素の合計を求めるメソッド
5、配列の要素の最大値を求めるメソッド
6、配列の要素の最小値を求めるメソッド
7、指定した値が配列内に存在するかどうかを調べるメソッド

解答例

```
import java.util.ArrayList;


public class Exercise2 {

	public static void main(String[] args) {
		double[] heightAry =
{185,184,179,181,171,176,182,178,179,177,187,178,175,173,178,181,181,175,180,184,1
77,177,173};

		Exercise2 ex = new Exercise2();
		for(int i = 0; i < heightAry.length; i++){
			ex.add(heightAry[i]);
		}

		System.out.println("合計値"+ex.sum());
		System.out.println("平均値"+ex.average());
		System.out.println("最大値"+ex.max());
		System.out.println("最小値"+ex.min());
		System.out.println("176があるかどうか："+ex.hasElement(176));
	}

	ArrayList<Double> ary;

	/**
	 * デフォルトコンストラクタ
	 *
	 */
```

```
	public Exercise2(){
		ary = new ArrayList<Double>();
	}

	/**
	 * 要素を追加するメソッド
	 * @param val
	 */
	public void add(double val){
		ary.add(val);
	}

	/**
	 * 要素の合計を求めるメソッド
	 */
	public double sum(){
		double sum = 0;
		for(int i = 0; i < ary.size(); i++){
			sum += ary.get(i);
		}
		return sum;
	}


	/**
	 * 要素の平均を求めるメソッド
	 */
	public double average(){
		double sum = 0;
		for(int i = 0; i < ary.size(); i++){
			sum += ary.get(i);
		}
		return sum/ary.size();
	}

	/**
```

```
	 * 要素の最大値を求めるメソッド
	 */
	public double max(){
		double max = -999999;
		for(int i = 0; i < ary.size(); i++){
			if(max < ary.get(i)){
				max = ary.get(i);
			}
		}
		return max;
	}

	/**
	 * 要素の最小値を求めるメソッド
	 */
	public double min(){
		double min = 999999;
		for(int i = 0; i < ary.size(); i++){
			if(min > ary.get(i)){
				min = ary.get(i);
			}
		}
		return min;
	}

	/**
	 * 値を要素に持っているかどうかを確認するメソッド
	 */
	public boolean hasElement(double d){
		return ary.contains(d);
	}

}
```

10.4

ある英語の文章中に同じ単語がいくつ含まれているかを調べたいと思います．HashMap を使ってこれを実現するクラスを作成しましょう．

ヒント

1、 単語をキーとして，その出現回数をデータとした HashMap を作成します．

2、 入力文章を String クラスの split メソッドによって単語に分割します．

3、 HashMap の中に単語が存在しなければ HashMap にキーとして単語を，データとして単語出現回数の 1 を入力します

4、 HashMap の中に単語が存在すれば HashMap からデータを取得して，1 追加した値を新たに入力します．

以下にプログラム例を示します．

```
import java.util.HashMap;
import java.util.Set;

public class Exercise3 {

	HashMap<String, Integer> wordMap; //単語と出現回数を記録する HashMap
	public Exercise3(){
		wordMap = new HashMap<String, Integer>();
	}
	//単語の出現回数を調べるメソッド
	public void add(String text){
		text = text.replaceAll("¥¥p{Punct}", ""); // (注)
		String[] words = text.split(" "); //単語の分割
		for(int i = 0; i < words.length; i++){
			//全ての単語に対して，出現したら 1 ずつ出現回数を増やします
		}
	}

	public void show(){
		//ここで，すべての単語とその出現回数を表示します．
	}
```

```
}
```

(注)この業では，特殊な処理として「.」や「,」といった単語を構成しない文字を削除しています．本書での説明範囲を超えているため，詳しい説明は省きます．興味がある方は，正規表現およびjava.util.regexパッケージにあるPattern,Matherについて調べてみてください．

解答例

```
import java.util.HashMap;
import java.util.Set;



public class Exercise3 {

	HashMap<String, Integer> wordMap;

	public Exercise3(){
		wordMap = new HashMap<String, Integer>();
	}

	public void add(String text){
		text = text.replaceAll("¥¥p{Punct}", ""); //.や,など単語構成後以外を削除
		String[] words = text.split(" ");
		for(int i = 0; i < words.length; i++){
			if(!wordMap.containsKey(words[i])){
				wordMap.put(words[i], 1);
			}
			else{
				int num = wordMap.get(words[i]);
				num++;
				wordMap.put(words[i], num);
			}
		}
	}
```

```
        public void show(){
                Set<String> wordSet = wordMap.keySet();
                for(String word: wordSet){
                        System.out.println(word+":"+wordMap.get(word));
                }
        }

}
```

利用例：

```
        public static void main(String[] args) {
                Exercise3 ex = new Exercise3();
                ex.add("This is a pen. This is an apple");
                ex.show();
        }
```

実行結果：

```
is:2
pen. :1
apple:1
an:1
a:1
This:2
```

11 変数とメモリ管理

11.1

以下の説明の中で正しい記述に○，間違っている記述に×を付けてください.

1、プリミティブ型の代入は値のコピーが行われる
2、参照型変数の初期値は null である
3、ローカル変数は初期化しなければ使えない
4、参照型同士を==で比較した場合，同一インスタンスかどうかを比較する
5、final 修飾子のついたフィールドを定数と呼ぶ

解答

1、○

2、○

3、○

4、○

5、×

static 修飾子と final 修飾子がついたフィールドを定数と呼びます.

11.2

以下のプログラムにおいて，a，b の値はそれぞれ何になるか述べてください.

```
import java.util.ArrayList;

public class Exercise2 {

	static public void main(String[] args){

		String a = "野球";

		String b = "サッカー";

		ArrayList<String> sportsList = new ArrayList<String>();

		sportsList.add(a);

		sportsList.add(b);

		//sportsList の最初の要素を変更

		sportsList.set(0, "テニス");

		//favoriteSportsList に sportsList を代入し，添え字 1 の要素を変更

		ArrayList<String> favoriteSportsList = sportsList;

		favoriteSportsList.set(1, "スノーボード");

		//b を sportsList の添え字 1 の要素に変更

		b = sportsList.get(1);


		//a と b の値は・・・？

		System.out.println("a="+a);

		System.out.println("b="+b);


	}
```

```
}
```

解答

a=野球

b=スノーボード

解説

a の値を sportsList の最初に代入して，これを，

```
sportsList.set(0, "テニス");
```

とテニスに変更しました．a の値もテニスに変更されそうですが，しかしながら，a が指している文字列は a に直接他の文字列を代入しない限り変化しません．

また， favoriteSportsList は sportsList のコピーの様に見えますが，実際は代入しているだけなので，同じオブジェクトを指しています．

そのため, favoriteSportsList の添え字 1 の要素を変更すると sportsList の添え字 1 の要素も変化し(favoriteSportsList と sportsList が同じモノだから当たり前ですが), sportsList の添え字 1 の要素を代入した b の値もスノーボードになります．

11.3

以下のプログラムの間違っている点を修正してください．

```
public class Exercise3 {
	static public void main(String[] args){
		int price = 100;

		if(price > 1000){
			String message = "お金がいっぱい";
		}else{
			message = "お金が少し";
		}
		System.out.println(message);
	}
}
```

解答

ローカル変数は，宣言したブロックでしか利用することが出来ません.

したがって，文字列 message は，

```
if(price > 1000){
        String message = "お金がいっぱい";
}
```

この範囲でしか利用できないのです.

message を if 文以降全てで使おうと思ったら，if 文の前にあらかじめ宣言しておく必要があります.

```
public class Exercise3 {
    static public void main(String[] args){
        int price = 100;

        String message;
        if(price > 1000){
            message = "お金がいっぱい";
        }else{
            message = "お金が少し";
        }
        System.out.println(message);
    }
}
```

11.4

以下のプログラムは，ある試験を行ったときに，合格点（60 点）以上を取った人の得点だけ取得することを目的として作成したものです．メソッド checkScore は引数 allScoreList で与えられた全員の得点から，60 点以上のものだけを残して，残りを削除しようとしています．しかし，実行した結果，うまく動いていないことがわかりました.

うまく動くように修正してください.

```
import java.util.ArrayList;
import java.util.Arrays;
```

```
public class Exercise4 {

	void checkScore(ArrayList<Integer> allScoreList){
		ArrayList<Integer> passList = new ArrayList<Integer>();
		for(int i = 0; i < allScoreList.size(); i++){
			if(allScoreList.get(i) >= 60){
				passList.add(allScoreList.get(i));
			}
		}
		allScoreList = passList;
	}
}
```

実行用プログラム

```
	static public void main(String[] args){
		Exercise4 ex = new Exercise4();

		ArrayList<Integer> scoreList = new ArrayList<Integer>();
		scoreList.add(82);
		scoreList.add(40);
		scoreList.add(63);
		scoreList.add(100);
		scoreList.add(75);
		scoreList.add(65);

		ex.checkScore(scoreList);

		for(int i = 0; i < scoreList.size(); i++){
			System.out.println(scoreList.get(i));
		}
	}
```

実行結果：

```
82
40
```

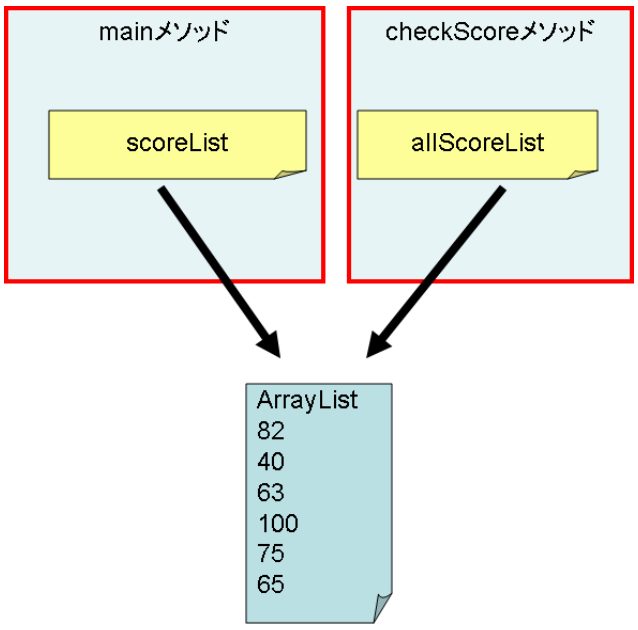
```
63
100
75
65
```

解答例

この例だと，新しく ArrayList インスタンス passedList を作成し，60 以上の値を passedList に挿入しています. そして,最後に変数 allScoreList に passedList を代入しています.

しかしながら，passedList を allScoreList 変数に代入しても，main メソッドで指していた socreList のインスタンスは変更されません. allScoreList 変数が passedList のオブジェクトを指すようになるだけで,，main メソッドの socreList のオブジェクトには影響を与えないのです.

そこで，新しくインスタンスを作るのではなく，直接 allScoreList インスタンスの中身を変更するようにプログラムを変更します．checkScore メソッド内の allScoreList の指すオブジェクトと main メソッドの socreList の指すオブジェクトは同じモノですので，checkScore メソッド内の allScoreList を変更すれば main メソッドの socreList も変更されます.




```
public class Exercise4 {
        void checkScore(ArrayList<Integer> allScoreList){
                for(int i = 0; i < allScoreList.size(); i++){
```

```
                    if(allScoreList.get(i) < 60){
                        allScoreList.remove(i);
                    }
                }
            }

}
```

ポイントは，「60 点以上を残す」のではなくて，「60 点未満を削除する」という作業を行なうとところです.

12　Java の高度な機能

12.1

以下の説明の中で正しい記述に○，間違っている記述に×を付けてください.

1、 try ブロックの後には必ず catch ブロックが必要である.
2、 RuntimeException から派生した例外も Exception として catch できる
3、 どのメソッドからも Exception から派生した例外が投げられる可能性がある
4、 throws 記述があるメソッドを使う場合は，必ず try～catch で囲まなければ行けない
5、 RuntimeException 例外が発生してもプログラムは停止しない
6、 finally ブロックは例外の有無にかかわらず必ず実行される

解答

1、○
2、○

以下のように書くことで，すべての例外を問答無用でキャッチすることができます.

```
        try{
            //Exception が発生する可能性がある処理
        }
```

```
        catch(Exception e){
                //Exception の処理
        }
```

3、×

Exception から派生した例外を投げるメソッドには必ず throws で，どのような例外が投げられるかを書く必要があります.

4、×

上位のメソッドでも throws 記述すれば try～catch ブロックで囲む必要はありません. また，throws に記述されている例外クラスが RuntimeException から派生したクラスだった場合も try～catch ブロックで囲む必要はありません.

5、×

try～catch ブロックで処理をしていない限りプログラムは停止します.

6、○

12.2

任意の英語文章ファイルを読み込んで単語の出現頻度を数えてください．なお，単語出現頻度計算には，問題 10.4 の単語数カウントクラスを利用します.

```
import java.util.HashMap;
import java.util.Set;

public class Exercise4 {
        HashMap<String, Integer> wordMap;

        public Exercise4(){
                wordMap = new HashMap<String, Integer>();
        }

        public void add(String text){
                text = text.replaceAll("¥¥p{Punct}", "");
                String[] words = text.split(" ");
                for(int i = 0; i < words.length; i++){
                        if(!wordMap.containsKey(words[i])){
                                wordMap.put(words[i], 1);
                        }
```

```
                    else{
                        int num = wordMap.get(words[i]);
                        num++;
                        wordMap.put(words[i], num);
                    }
                }
        }

        public void show(){
                Set<String> wordSet = wordMap.keySet();
                for(String word: wordSet){
                        System.out.println(word+":"+wordMap.get(word));
                }
        }

}
```

解答例

以下のように実装すれば，単語数をカウントできます.

ただし，単語頻度計算用のクラス名は WordCounter です.

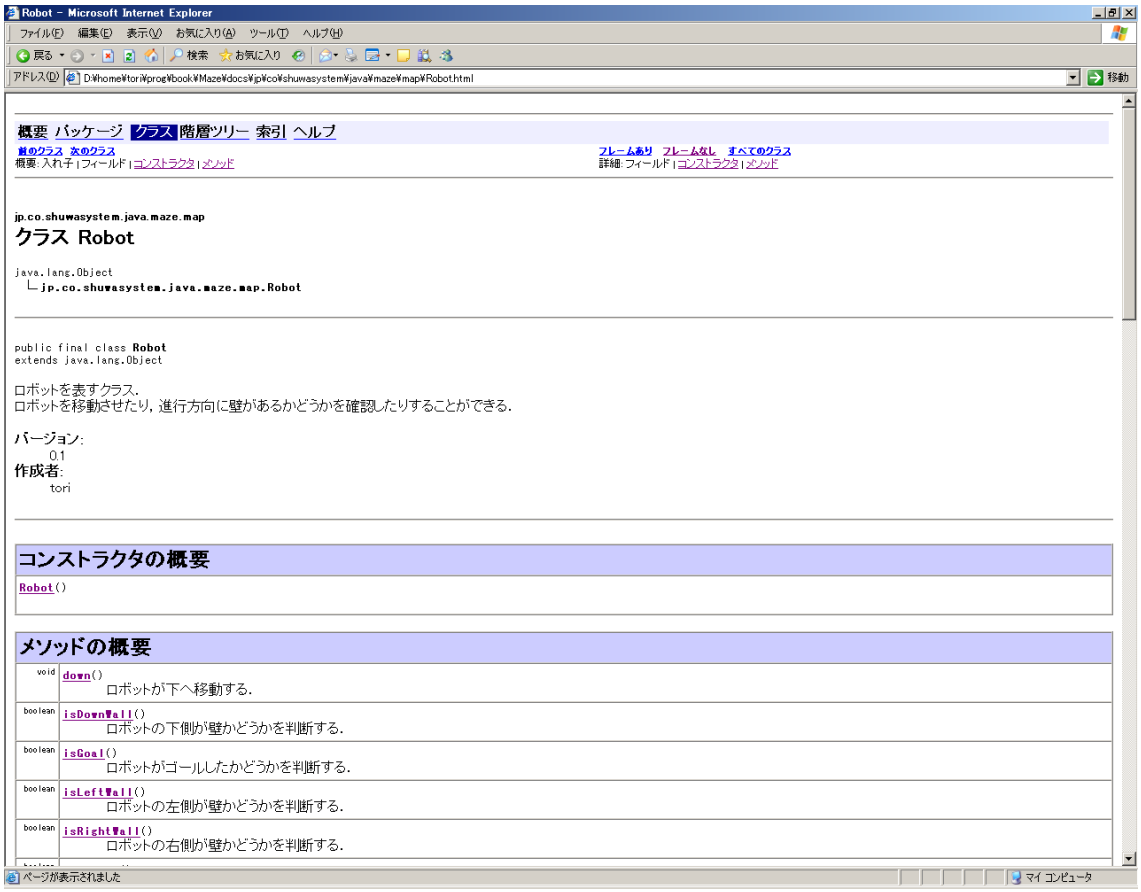
```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;

public class Exercise2 {
        public static void main(String[] args) {
                String fileName = args[0];
                WordCounter wordCounter = new WordCounter();
                try{
                        Reader reader = new FileReader(fileName);
                        BufferedReader br = new BufferedReader(reader);
                        String text = null;
                        while((text = br.readLine()) != null){
                                wordCounter.add(text);
                        }
```

```
                    br.close();
                    reader.close();
                }catch(IOException e){
                    e.printStackTrace();
                }
                wordCounter.show();
        }
}
```

12.3

URL に Robot クラスの JavaDoc があります．この JavaDoc を見ながら，User.java を改良して任意の迷路でロボット君をゴールに到達できるようにプログラミングしてください．


Robot - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
戻る 検索 お気に入り
アドレス(D) D:¥home¥tori¥prog¥book¥Maze¥docs¥jp¥co¥shuwasystem¥java¥maze¥map¥Robot.html
移動
概要 パッケージ クラス 階層ツリー 索引 ヘルプ
前のクラス 次のクラス
概要: 入れ子 | フィールド | コンストラクタ | メソッド
フレームあり フレームなし すべてのクラス
詳細: フィールド | コンストラクタ | メソッド
jp.co.shuwasystem.java.maze.map
クラス Robot
java.lang.Object
└ jp.co.shuwasystem.java.maze.map.Robot
public final class Robot
extends java.lang.Object
ロボットを表すクラス.
ロボットを移動させたり, 進行方向に壁があるかどうかを確認したりすることができる.
バージョン:
0.1
作成者:
tori
コンストラクタの概要
Robot()
メソッドの概要
void down()
ロボットが下へ移動する.
boolean isDownWall()
ロボットの下側が壁かどうかを判断する.
boolean isGoal()
ロボットがゴールしたかどうかを判断する.
boolean isLeftWall()
ロボットの左側が壁かどうかを判断する.
boolean isRightWall()
ロボットの右側が壁かどうかを判断する.
ページが表示されました
マイ コンピュータ