ソフトウェア工学
第8回

土田　隼之

週	授業内容・方法	週ごとの到達目標
1週	ソフトウェアの性質と開発	ソフトウェア開発の特徴および課題について少なくとも一つ上げられ、その理由を言える。
2週	ソフトウェア開発プロセス	複数の開発プロセスモデルを挙げ、それぞれの特徴を言える。
3週	要求分析	要求分析とプロトタイピングの関係性や有用性について言える。
4週	ソフトウェア設計	モジュールの結合度の低い場合と高い場合のモジュール間の依存性について述べ、モジュール結合度の低い具体例を言える。
5週	プログラミングとテスト	誤り混入をさせないためのプログラミング手法およびテスト効率を向上させる技法について言える。
6週	テストと保守	保守容易性を確保するための方策について、考察し、述べることができる。
7週	グループワーク	前半6週に関する課題を、グループワークで取り組む。
8週	中間試験	前半に習得した項目について確認する。
9週	オブジェクト指向１	身の
10週	オブジェクト指向２	オブ
11週	ソフトウェア再利用	ソフ
12週	プロジェクト管理	プロ
13週	品質管理	品質
14週	ソフトウェア開発規模と見積もり	ソフトウェア開発規模の見積もり手法について言える。
15週	グループワーク	後半6週に関する課題を、グループワークで取り組む。
16週	期末試験	後半に習得した項目について確認する。

9週:中間試験を実施予定
8週:中間試験の模擬問題を予定
→傾向を見てもらう目的。模擬問題はポートフォリオ点に入るので、試験点には影響なし。#課題をきちんと出してれば、模擬試験0点でもポートフォリオ点満点

今日の内容

1)テストの追加説明

2)模擬試験

3)オブジェクト指向

1)+2)45分

1)10分
2)15分+10分+10分
3)45分

ホワイトボックステスト

過去の講義

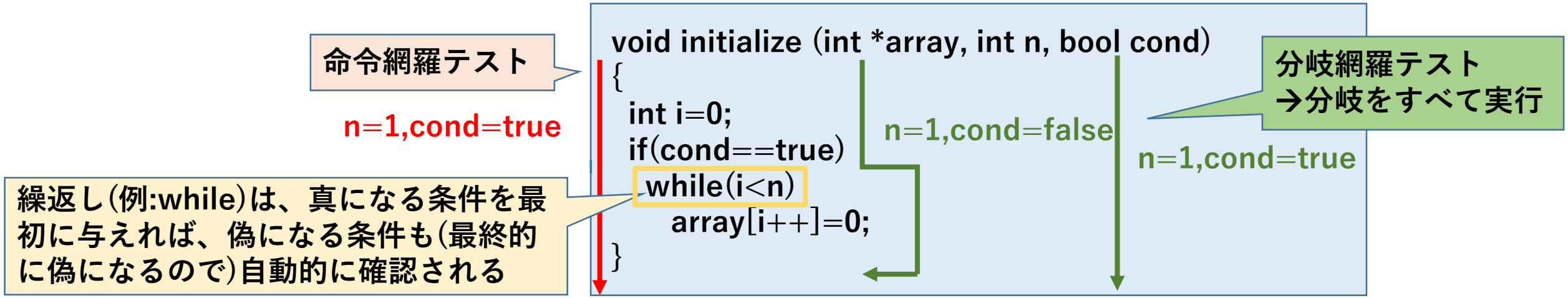
- プログラムの内部処理に注目し、プログラム構造を網羅するようにテストケースを見いだす方法である。

a)命令網羅テスト:プログラムの全ステートメントが実行されているかどうかを網羅度の基準とする。この基準を命令網羅と呼ぶ。

例:下記では「n=1,cond=true」で網羅度を達成できる。

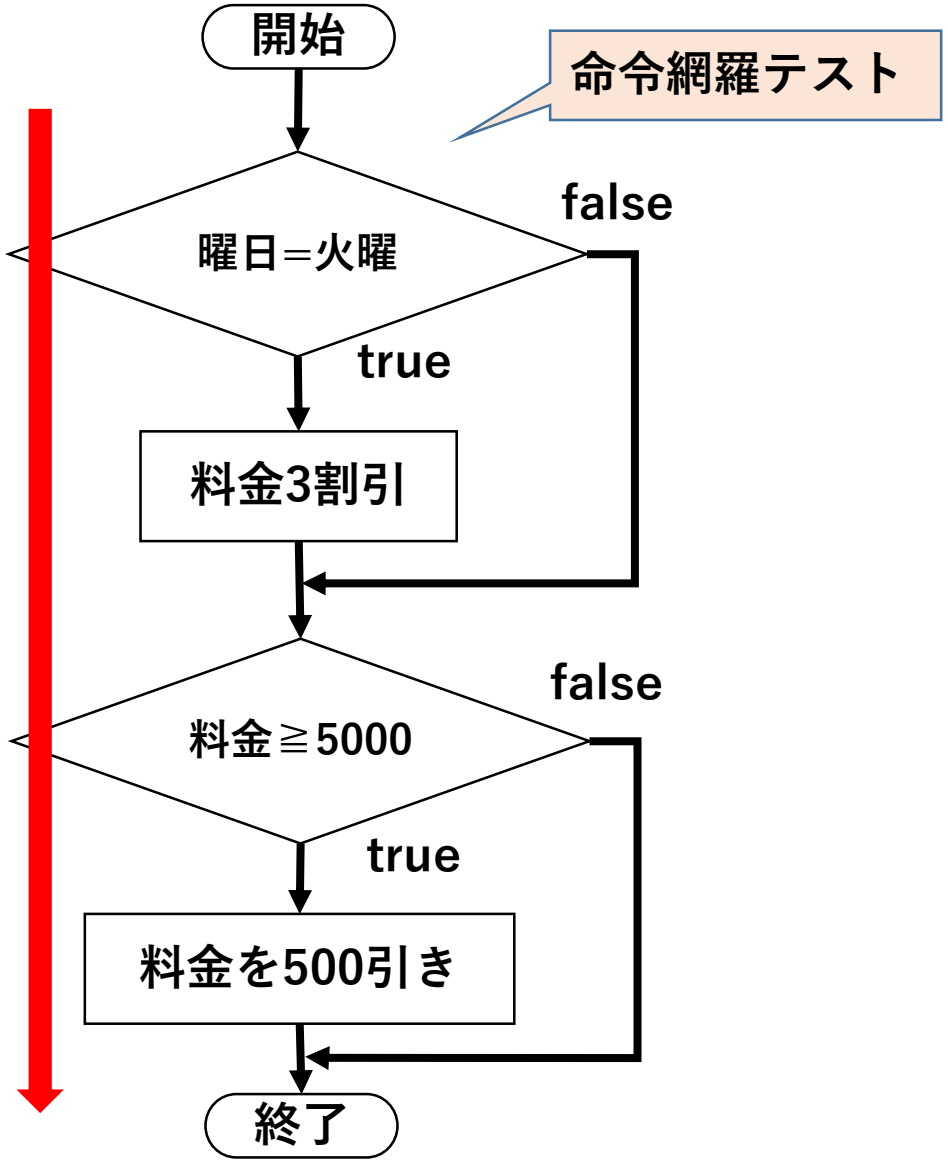
b)分岐網羅テスト:プログラムの条件分岐がすべて実行されているかどうかを網羅度の基準とする。この基準を分岐網羅と呼ぶ。

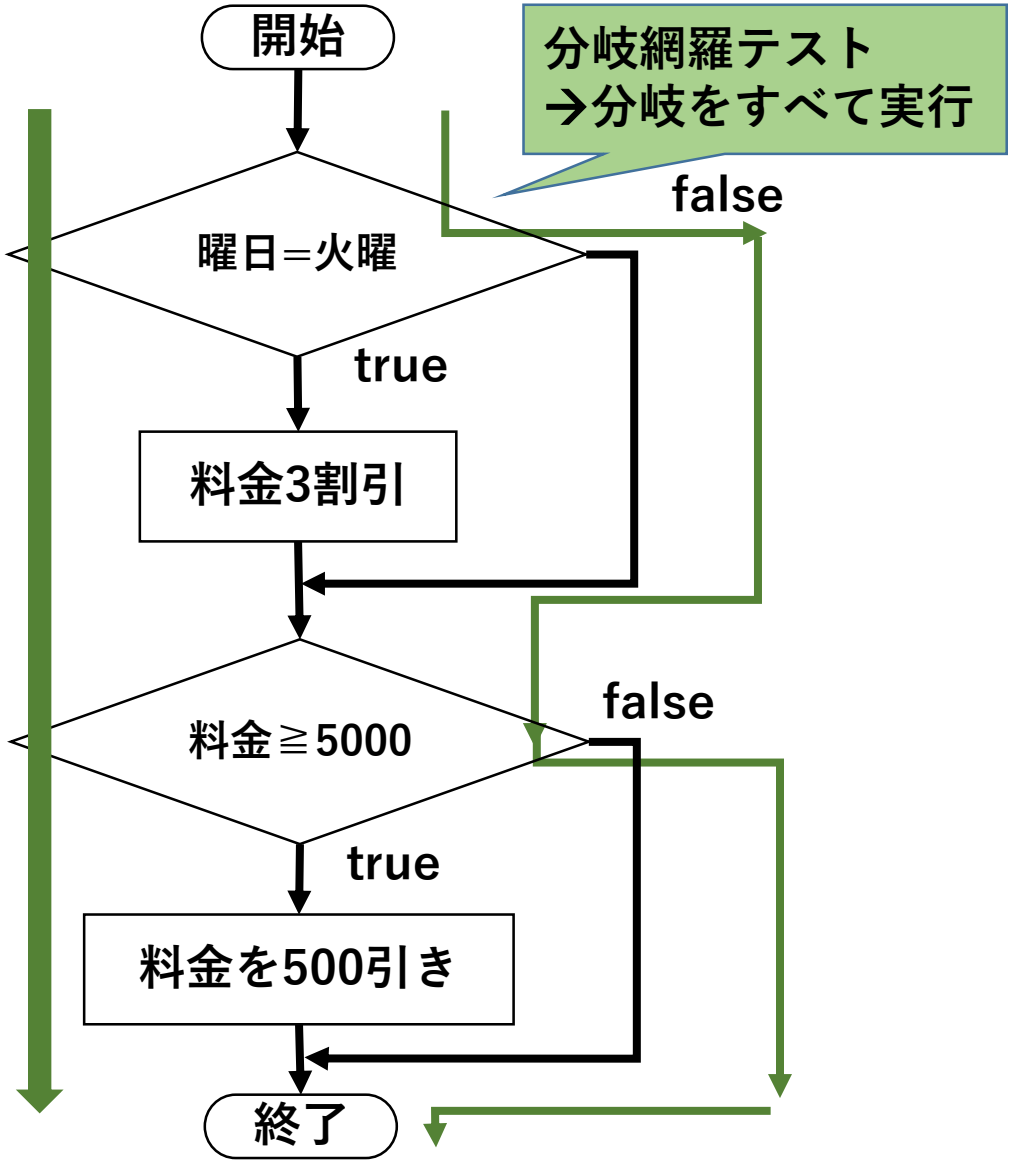
例:下記では「n=1,cond=true」「n=1,cond=false」で網羅度達成できる。



ホワイトボックステスト(命令網羅・分岐網羅)

過去の講義



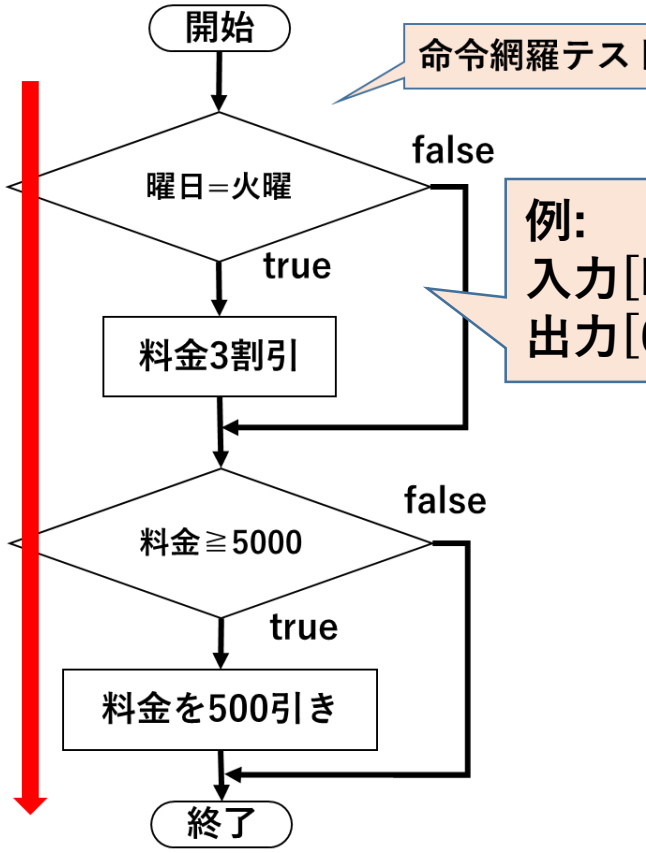


課題6-1(締切:6/9)

前回の講義

前スライドの処理について、命令網羅と分岐網羅で必要なテスト条件を記載してください。なお、テスト条件は(入力:料金,曜日 出力:割引後料金)としてください。

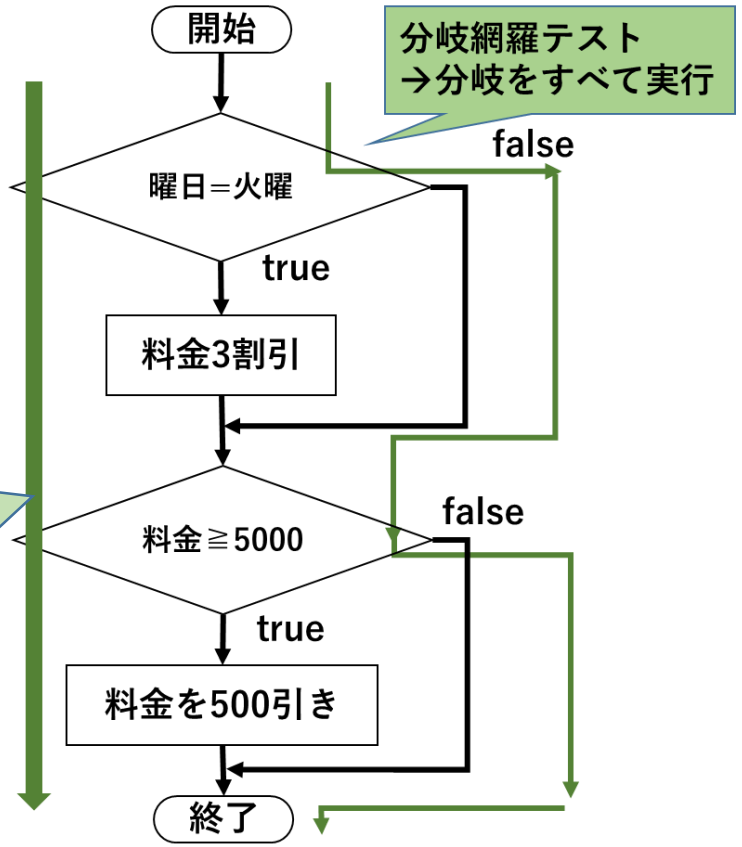
まずは、1)エクセルなどの表で作成し、その後に2)python実装して「ソースコードのテキスト」と「テスト結果のスクリーンショット」を提出ください。提出:1)2)の両方を提出ください



例:
入力[曜日:火曜、料金10000円]
出力[6500円(=10000×0.7-500)]

例:
テスト条件1)
入力[曜日:火曜、料金10000円]
出力[6500円(=10000×0.7-500)]

テスト条件2)
入力[曜日:水曜、料金3000円]
出力[3000円]



ホワイトボックステスト2(条件網羅)

1つのif文が複数の式から構成される場合、それぞれの式に対する真と偽のテストケースを用意する必要がある。

c)条件網羅テスト:プログラムの条件分岐を構成する各式に対し真と偽が網羅されているかどうかを網羅度の基準とする。この基準を条件網羅と呼ぶ。
例:下記では「n=1,cond=true,flag=2」「n=2,cond=false,flag=0」で網羅度達成できる。**分岐網羅を満たさない場合がある事に注意されたい。**

```
void initialize (int *value, int n, bool cond, int flag){
  if(cond==true&&flag>1){
   printf("flag on¥n");
  }
 else printf("flag off¥n");

  if(n%2==0||flag==2){
        *value=*value*2;
 }
 else *value=*value*n;

}
```

cond=true,flag=2
cond=false,flag=0

n=2,flag=2
n=1,flag=0

cond==trueとflag>1の
それぞれについて
真偽のテストケースを用意
#組合せは考慮不要

判定条件1
cond==trueの真(cond=true)と偽(cond=false)
flag>1の真(flag>1)と偽(flag<=1)
→cond=true,cond=false,flag>1,flag<=1を満たす必要

判定条件2
n%2==0の真(n=2*x)と偽(n=2*x+1) xは整数
flag==2の真(flag=2)と偽(flag≠2)
→n=2*x,n=2*x+1,flag=2,flag≠2を満たす必要

ホワイトボックステスト3(複合条件網羅)

d)複合条件網羅テスト:プログラムの条件分岐を構成する各式の真偽値のすべての組合せが網羅されているかどうかを網羅度の基準とする。この基準を複合条件網羅と呼ぶ。

```
void initialize (int *value, int n, bool cond, int flag){
  if(cond==true&&flag>1){
    printf("flag on¥n");
  }
  else printf("flag off¥n");

  if(n%2==0||flag==2){
        *value=*value*2;
  }
  else *value=*value*n;
}
```

cond=true,flag=2
cond=false,flag=0

n=2,flag=2
n=1,flag=0

判定条件1
1-A)cond==trueの真(cond=true)と偽(cond=false)
1-B)flag>1の真(flag>1)と偽(flag<=1)
→cond=true,cond=false,flag>1,flag<=1を満たす必要

cond==trueとflag>1のそれぞれについて真偽のテストケースを用意
#組合せ網羅が必要

判定条件2
2-A)n%2==0の真(n=2*x)と偽(n=2*x+1) xは整数
2-B)flag==2の真(flag=2)と偽(flag≠2)
→n=2*x,n=2*x+1,flag=2,flag≠2を満たす必要

	1-A)	1-B
1	真(cond=true)	真(flag=2)
2	真(cond=true)	偽(flag=0)
3	偽(cond=false)	真(flag=2)
4	偽(cond=false)	偽(flag=0)

	2-A)	2-B
5	真(n=2)	真(flag=2)
6	真(n=2)	偽(flag=0)
7	偽(n=1)	真(flag=2)
8	偽(n=1)	偽(flag=0)

2,3,6,7を満たすテストケースを追加すると複合条件網羅となる

判定1,判定2のそれぞれの中で網羅されていれば良い

ブラックボックステスト(同値分割)

過去の講義

- プログラムの機能記述(仕様)に基づき、可能な入力の組合せとその入力に対する出力を選択する方法である。

a)同値分割:プログラムの入力条件を使って、テスト入力空間を分割する方法である。プログラムにとって同じ扱いを受けるはずの値の範囲である同値クラスという概念を考慮する。プログラムにとって有効な入力を同値有効クラス、無効なものを無効同値クラスと呼ぶ。

例えば、以下の要求仕様について考える。

入力A:1～99まで入力可能、入力B:1～99まで入力可能、出力C:A×B

```
if(a>0&&a<=99)
   //正しい値が入力されたときの処理
else
  //間違った値が入力されたときの処理
```

境界値(例:0,1,99,100)と代表値(例:範囲の中央値 49)

無効同値(～0) | 有効同値(1～99) | 無効同値(100～)

それぞれの代表値だけをテストすることで、テスト数増加を防ぐ

ブラックボックステスト2(限界値分析)

過去の講義

b)限界値分析:同値分割法では見つからないプログラミング上の実装ミスを見い出すための技法である。プログラム処理の変わり目と思われる値に焦点を絞り、テストケースを設計する。

例: if(input>0&&input<64){/*処理*/}で、正しくはinput<=64

例:ユーザがエディタの印刷機能を使う場合を考える。1ページ未満の印刷をユーザが要求した場合はエラーとする。

```
if(a>=1)
   //印刷機能
else
   //エラー処理
```

```
if(a>1)
   //印刷機能
else
   //エラー処理
```

```
if(a>=2)
   //印刷機能
else
   //エラー処理
```

```
if(a>=1)
   //印刷機能
/*else
   //エラー処理
*/
```

```
if(a>=1&&a<10)
   //印刷機能
/*else
   //エラー処理
*/
```

ブラックボックステスト2(限界値分析)

過去の講義

例:ユーザがエディタの印刷機能を使う場合を考える。1ページ未満の印刷をユーザが要求した場合はエラーとする。

```
if(a>=1)
   //印刷機能
else
   //エラー処理
```

```
//タイプ1のバグ
if(a>1)
   //印刷機能
else
   //エラー処理
```

```
if(a>=2)
   //印刷機能
else
   //エラー処理
```

```
if(a>=1)
   //印刷機能
/*else
   //エラー処理
*/
```

```
if(a>=1&&a<10)
   //印刷機能
/*else
   //エラー処理
*/
```

正しいテストケース例:

テストケース1:1を入力

　結果:正常な処理がなされること

テストケース2:0を入力

　結果:エラー処理がなされること

適切でないテストケース例:
テストケース1:-1を入力
テストケース2:2を入力
とすると、タイプ1のバグが見つからない

異なる処理が行われる一番近い二地点をテストケースとする

課題6-2(締切:6/9)

前回の講義

ある動物飼育用の室温計

室温計は現在の室温表示欄と、ある動物にとって適温か表示するメッセージ欄がある。室温に応じて以下のメッセージが表示される。表示メッセージのpythonのテストケースを同値分割と限界値分析を用いて作成ください。なお、「表示メッセージは関数返り値」としてください。

前提:室温計の精度は0.1°C単位とする

室温計で計測できる下限および上限を考慮する必要はないとする。

No	室温	表示メッセージ
1	23.0°C未満	寒い
2	23.0°C以上25°C未満	快適
3	25°C以上	暑い

寒い、快適、熱いの
「境界値(下限)、代表値、境界値(上限)」
をテストケースとする。
#境界値の下限、上限は無い場合もある

入力(室温)	出力(表示メッセージ)	
20.0℃	寒い	寒いの代表値
22.9℃	寒い	寒いの境界値
23.0℃	快適	快適の境界値
24.0℃	快適	快適の代表値
24.9℃	快適	快適の境界値
25.0℃	暑い	暑いの境界値
30.0℃	暑い	暑いの代表値

今日の内容

1)テストのおさらい

2)模擬試験

3)オブジェクト指向

2)模擬試験

注意点:
グループワークは、履修届のある者を無作為に抽出し複数名でグループを作り、作業を行うため、グループワークの日はもとよりそれまでの授業についても欠席しないよう注意されたい。
合格の対象としない欠席条件(割合) 1/3以上の欠課

	試験	グループワーク	相互評価	態度	ポートフォリオ	その他	合計
総合評価割合	60	20	0	0	20	0	100
基礎的能力	0	0	0	0	0	0	0
専門的能力	60	20	0	0	20	0	100
分野横断的能力	0	0	0	0	0	0	0

1年生向け講義アイデアの具体的な資料を作成してもらいます
→出されてないと0点もある

課題とグループワーク成果物
→出されてないと0点もある

テストケースを作成する問題などを予定してます。
→模擬試験で傾向説明
出題範囲:テスト、UML関連(例:間違い探し、自分で書く)
明日、明後日で、出題範囲のスライドを連絡します。

今日の内容

1)テストのおさらい

2)模擬試験

3)オブジェクト指向

オブジェクト指向とは

- オブジェクト指向とは、もの(オブジェクト)を中心としてモデル化する考え方、あるいはその考え方に基づくシステム構成法を指す。
- オブジェクト指向はソフトウェア開発において高い作業効率性や再利用性を達成するための技術として活用されており、さらにプログラム自動生成や自動検証を実現する基盤技術としても期待されている。

オブジェクト指向とプログラム構成法

- プログラム構成法として、長らく機能中心のモジュール分割法(構造化設計)が用いられてきた。

→過度に機能自体の構造を反映する構成に導き、システムの保守性を欠くモジュール構造を作り出してしまったという反省があった。反省の中で、オブジェクト指向は情報隠蔽の最良の形式として発展してきた。

プログラムの修正(例:関数の機能変更)をするときに、プログラム全体ではなく一部分だけ理解して修正できるのが望ましい。→何も考えずにコーディングすると、例えば、グローバル変数はプログラムの広い領域から使われる可能性。オブジェクト指向の考え方で、影響範囲を限定できると嬉しい。
自分担当の10万行で整合性を確保と、他の50人がコーディングしている90万行含めての整合性確保だとだいぶん違う。

→より使いやすいプログラミング言語としてJavaやモデル記述言語(UML)へと発展してきている。

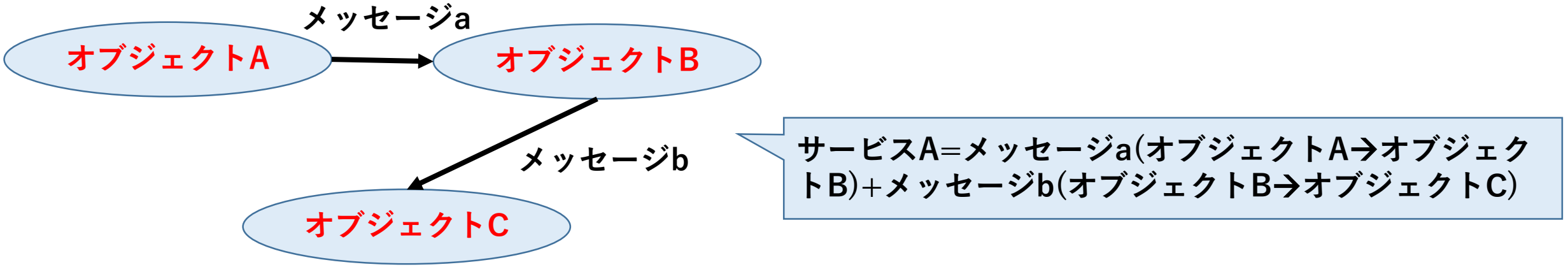
基本概念

- オブジェクト指向の基本構成要素は、オブジェクトとメッセージである。オブジェクト指向は、オブジェクトを唯一の存在とし、すべての出来事をオブジェクト同士がメッセージを送りあうことにより表現する考え方である。

オブジェクト:実在するものや概念

メッセージ:オブジェクト間の情報のやり取りや仕事の依頼

オブジェクトは、メッセージを受けられるように、識別可能になっており、受信可能なメッセージ形式が定義されている。

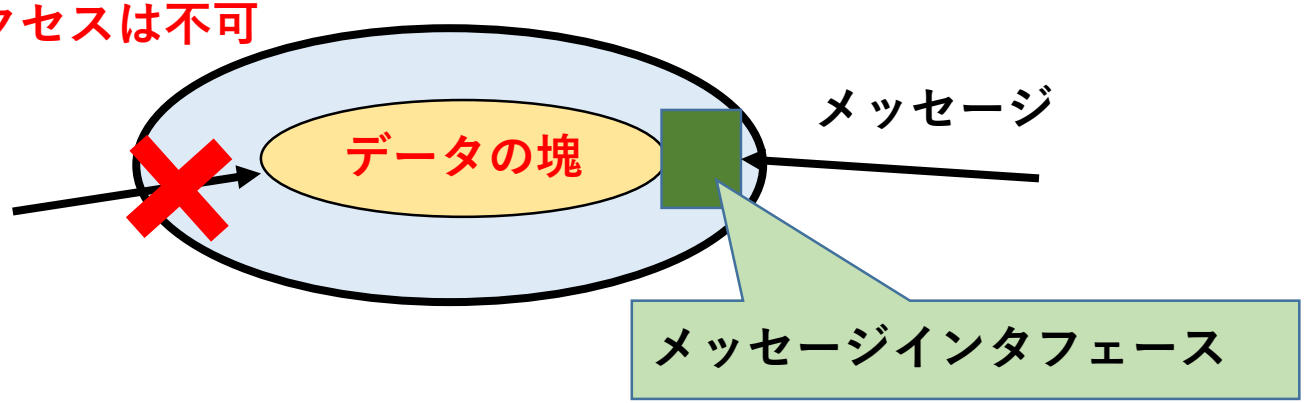


a)カプセル化

- カプセル化とは、オブジェクトのふるまいを「それを起動するメッセージインタフェース(仕様)」と「それが果たす機能の実現(実装)」とに分け、実装を隠すことである。
- カプセル化は、設計・プログラミングにおける抽象データ型に相当し、オブジェクトのモジュール性を向上する機構である。

抽象データ型の例:二分木
→二分木を構成する配列などに直接アクセスするのではなく、挿入、削除、探索などの関数経由で操作が可能

データへの直接アクセスは不可



オブジェクトは、メッセージインターフェースという殻に守られた卵のようなもの。

外からは殻しか見えず、殻が変化しなければ、中身(データと機能)に変化があっても外の世界は何の影響もうけない。

メッセージインタフェースだけ理解すればよく、実装は理解しなくて良いのでプログラムを理解するときに、効率的になる。

a)カプセル化2

- クラスとインスタンスは、オブジェクト指向におけるカプセル化を実現するための手段である。
- **クラス**:同じ特性を持つオブジェクトのグループの記述

クラスの例:「時計とは、現在時刻を持ち時刻指定ができるものである」

- **インスタンス**:クラスに対して実存するオブジェクトを意味する。

インスタンスの例:「私の時計」→固有データとして現在時刻を持っている。

→クラスは、インスタンスを作り出すためのひな型に相当する。
　クラスからインスタンスが作られる時、**具象化**されるという。

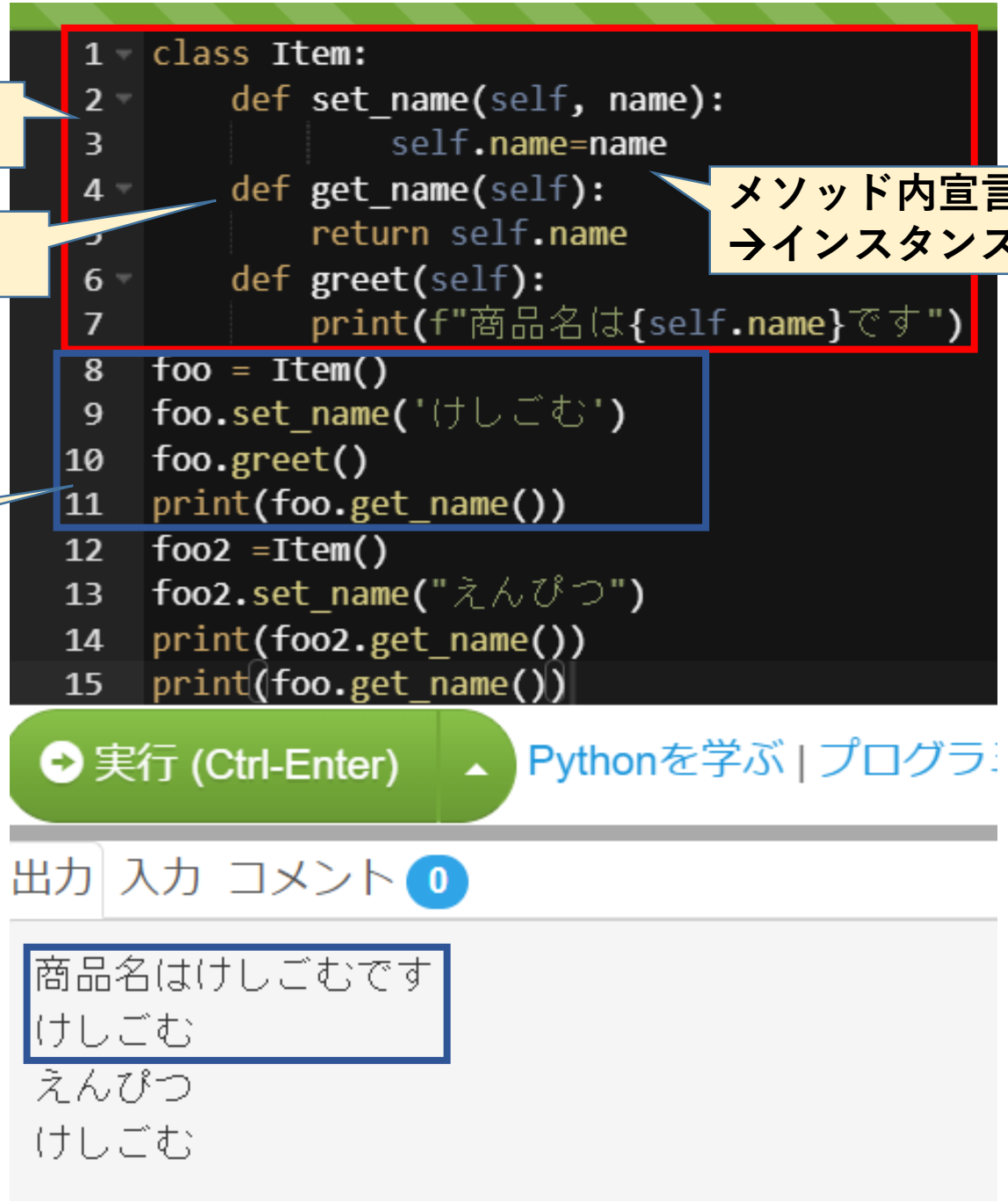
```
class Item:
    def set_name(self, name):
        self.name=name
    def get_name(self):
        return self.name
    def greet(self):
        print(f"商品名は{self.name}です")
foo = Item()
foo.set_name('けしごむ')
foo.greet()
print(foo.get_name())
foo2 =Item()
foo2.set_name("えんぴつ")
```

Itemクラスの定義

Itemクラスのメソッド定義

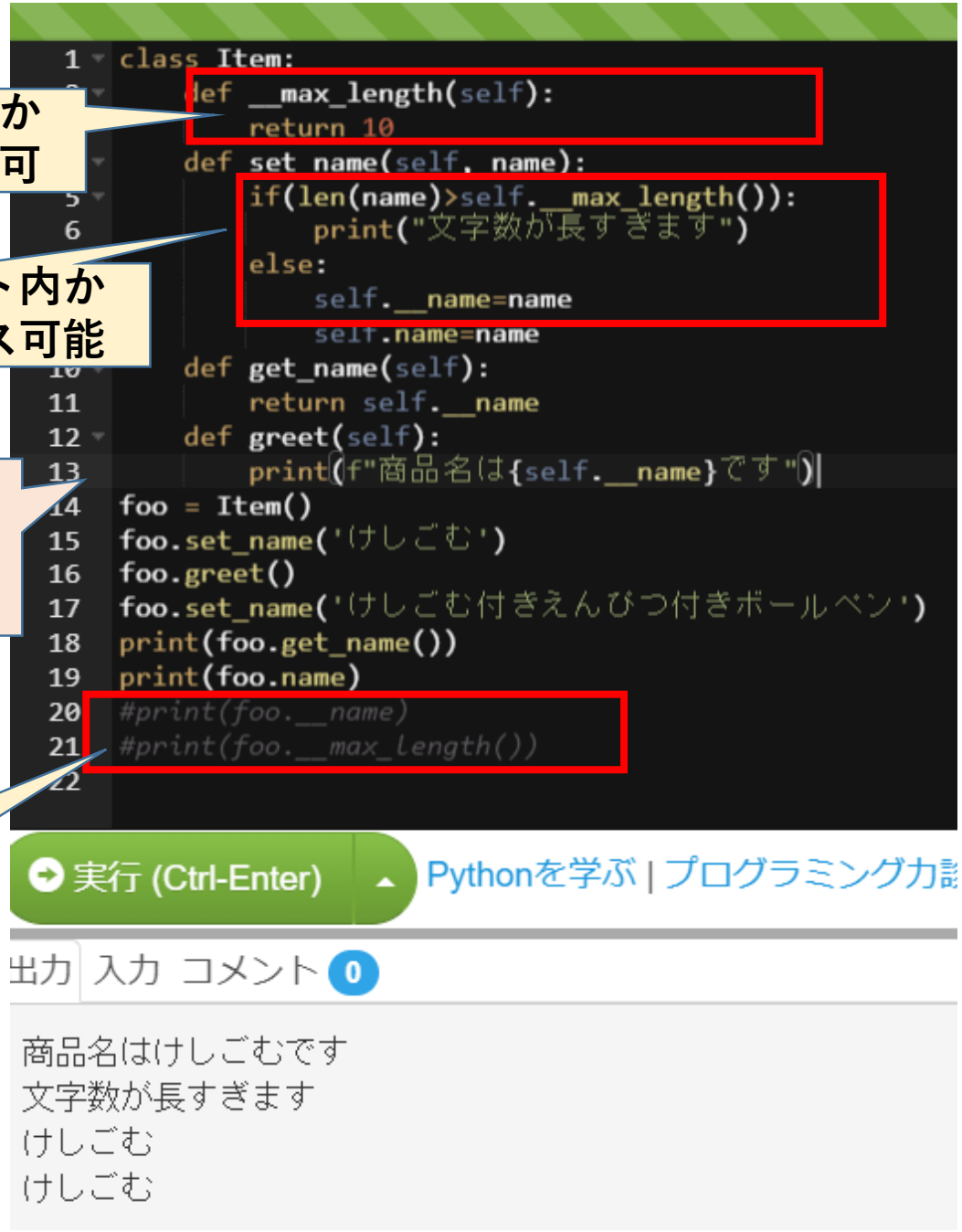
けしごむインスタンスの生成

メソッド内宣言
→インスタンス変数



```
class Item:
    def __max_length(self):
        return 10
    def set_name(self, name):
        if(len(name)>self.__max_length()):
            print("文字数が長すぎます")
        else:
            self.name=name
    def get_name(self):
        return self.name def greet(self):
        print(f"商品名は{self.name}です")
foo = Item()
foo.set_name('けしごむ')
foo.greet()
foo.set_name('けしごむ付きえんぴつ付きボールペン')
print(foo.get_name())
#print(foo.__max_length())
```

メソッド名やインスタンス変数名の前に__を付けるとオブジェクト外からのアクセス不可になる



オブジェクト外からはアクセス不可

オブジェクト内からはアクセス可能

オブジェクト外からはアクセス不可

b)ポリモルフィズム

- ポリモルフィズムは、本来、型のあるプログラミング言語において、型に束縛されずにアルゴリズムを再利用するための言語理論的な機構を指す言葉である。

関数呼び出し側の記載した、関数引数の型や個数に応じて、自動的に処理を変えてくれる。

- オブジェクト指向では、ポリモルフィズムはメッセージ送信において受信側のオブジェクトが動作を決定することを指す。

→メッセージ送信側のオブジェクトが、受信者が何もであるかを(深く)知らなくてよく、送信側の受信側に対する依存度が減る。ポリモルフィズムは、多種多様なオブジェクトを構成要素として保持するようなオブジェクトの再利用性を高める効果を持つ。

受け手側が「自分は何者か」で動作を判断

・メモ帳
Dolt→メモ帳を開く

・鉛筆
Dolt→文字を書く

デスクトップオブジェクト

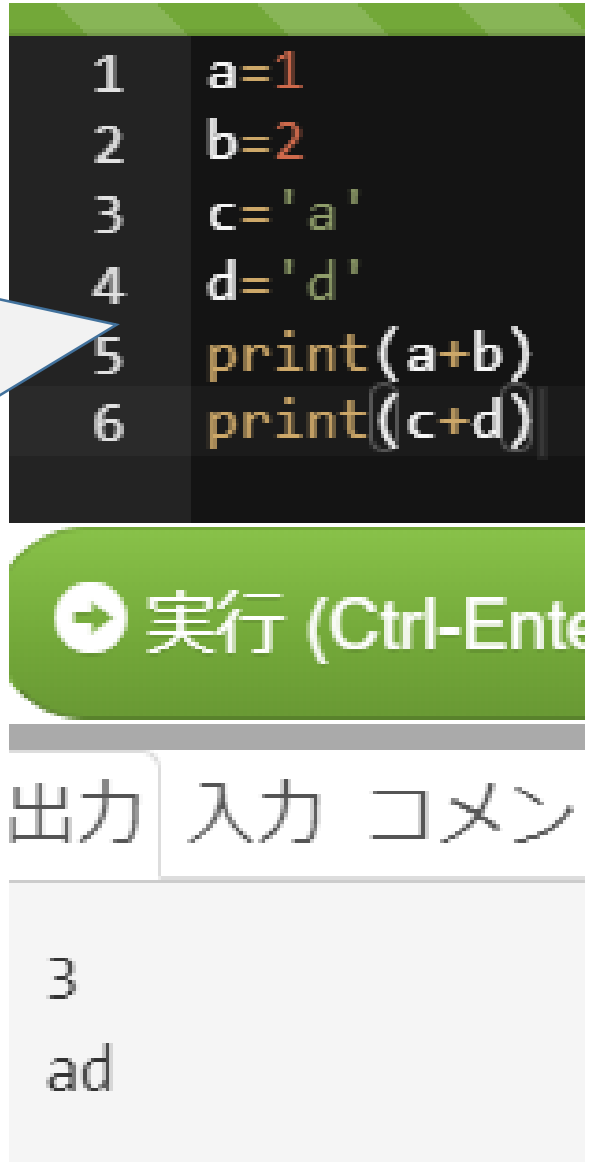
デスクトップは、デスクトップオブジェクトがDoltというメッセージインタフェースがあることだけを知っている
#複数クラスで、同一メソッドが利用可能

デスクトップ

素材利用:いらすとや

```
a=1
b=2
c='a‘
d='d‘
print(a+b)
print(c+d)
```

+演算子は、
1)操作対象の両方が整数の場合は加算を行い
2)捜査対象の両方が文字(列)の場合は文字連結を行う
→ポリモルフィズム
送信元は、受信側の処理メソッドを考慮しなくてよい。もし、文字列連結専用、数値加算専用のメソッドがあれば送信元は考慮の必要。



オブジェクト指向によるもの作り

- オブジェクト指向は、オブジェクト中心に対象世界をモデリングする考え方と、それによってもたらされる高い保守性・再利用性の２つに大きな特徴がある。

→オブジェクト指向を単にプログラミング段階でのみ適用したのでは、得ることが難しい。開発システムの保守性と再利用性はその問題領域の性質に強く依存し、上流過程である分析と設計で考慮される必要がある。

オブジェクト指向分析

- 要求分析とは、ユーザ要求、すなわちこれから作成するソフトウェアがどうあるべきかを、ユーザから獲得・表現・妥当性確認する工程である。
- 分析には、問題領域を理解することと問題の解を仕様化する2つの役割がある。

1)問題領域を理解する
→実世界をそのまま記述
　対象物と対象物間の関係、全体としてのふるまい、制約事項

実世界の対象物とそれらの関係を理解し、ユーザが望むシステムのふるまいは何かを把握し、それらを実現するうえでの制約を明らかにすること。

2)問題の解を仕様化する
→ユーザが望むシステムを記述
　システムの入出力、タイミング、ユーザインタフェース

ユーザが望むものを厳密な要求仕様としてまとめ、設計者に渡す

第8回課題(締切6/23)

課題)かんたんなプライベートなクラス変数と、クラスメソッドを定義してください。オブジェクト内からはアクセス可能であるが、オブジェクト外からはアクセス不可であることを確認して、1)確認した結果のスクリーンショットと2)ソースコードのテキストを提出ください。