ドラクエ風「商人クラス」で学ぶ getter / setter / public / private

☒ 授業進行カンペ＋教材一体版

ドラクエ風「商人クラス」で学ぶ getter / setter / public / private

☒ 授業概要

- 対象：専門学校1年（C++基礎中盤）
- 時間：45分 × 1コマ
- 前提：クラス宣言・メンバ関数は既習
- 目的：getter/setter・public/private の基本構造と目的を理解する
- 進行：
  1. バグる商人（問題提起）
  2. 安全な商人（getter/setter導入）
  3. 演習問題（小課題＋全体レビュー）

☒ 提示する資料（学生閲覧用 DokuWiki ページ）

BookStackからリンクまたはQRで提示。学生側にはDokuWiki版をコピー配布。

ページタイトル:

> ☒ ドラクエ風「商人クラス」で学ぶ getter / setter / public / private

構成:

- バグる商人コード（直接代入で壊れる例）
- 安全な商人コード（getter/setter＋private）
- 解説表：public／private／getter／setter
- 練習問題5問
- チェックポイント

提示タイミング:

- 導入5分で「このページを開いてください」
- 教員は1つずつ実行例を見せながら口頭解説

☒ 授業進行

時間	内容	教員アクション
0〜5分	導入：商人バグ実行	「やくそう−100G」など笑いを取りながら興味づけ
5〜15分	getter/setterの考え方	「値の出入口を関数で管理」と板書
15〜25分	安全な商人コード実行	private→public→setterの流れを追う
25〜35分	演習問題	1〜4を個人、5はグループ討論
35〜40分	全体レビュー	setterの制限・public/privateを再確認
40〜45分	まとめ＋次回予告	「次は勇者(Player)と取引します」

☒ 教員トーク例

> 「この商人、在庫2個で始めたのに、勇者が直接 `stock=9999` って書いたらどうなる？」
> 「マイナス価格にしたら勇者が儲かる！ これはバグ。
> だから金庫（private）に鍵をかけて、店員（getter/setter）を通すんです。」

☒ 教示コード（投影または配布用）

① バグる商人（問題提起）

```
#include <iostream>
#include <string>
using namespace std;

class Merchant {
public:
    string itemName;
    int price;
    int stock;

    void Sell() {
        if (stock > 0) {
            stock--;
            cout << itemName << "を" << price << "Gで売った！" << endl;
        } else {
            cout << "在庫がない！" << endl;
        }
    }
};

int main() {
    Merchant m;
    m.itemName = "やくそう";
    m.price = 8;
    m.stock = 2;

    m.Sell();
    m.Sell();
    m.Sell();

    m.stock = 9999; // ← 不正操作
    m.price = -100; // ← マイナス価格！

    m.Sell();
}
```

② 安全な商人（getter/setter + private）

```
#include <iostream>
#include <string>
using namespace std;

class Merchant {
private:
    string itemName;
    int price;
    int stock;

public:
    void SetItemName(string name) { itemName = name; }

    void SetPrice(int value) {
```

```
        if (value < 0) value = 0;
        if (value > 9999) value = 9999;
        price = value;
    }

    void SetStock(int value) {
        if (value < 0) value = 0;
        stock = value;
    }

    string GetItemName() { return itemName; }
    int GetPrice() { return price; }
    int GetStock() { return stock; }

    void Sell() {
        if (stock > 0) {
            stock--;
            cout << itemName << "を" << price << "Gで売った！" << endl;
        } else {
            cout << "在庫がない！" << endl;
        }
    }
};

int main() {
    Merchant m;
    m.SetItemName("やくそう");
    m.SetPrice(8);
    m.SetStock(2);

    m.Sell();
    m.Sell();
    m.Sell(); // 在庫がない！

    // m.stock = 9999; ← エラー
    // m.price = -100; ← エラー
}
```

☒ 授業内演習問題（提示用）

Q1

商人クラスに「商品名」「価格」「在庫」を **private** で定義し、
それぞれに `Set` ／ `Get` 関数を作成せよ。

Q2

`SetPrice()` で価格が **0未満なら0、9999より大きければ9999** に制限せよ。

Q3

`SetStock()` で在庫が **マイナスのとき0に補正** されるようにせよ。

Q4

`main()` 関数で商人を作成し、やくそうを2個販売するコードを書け。
（3回目の販売時には「在庫がない！」が出力されること）

Q5（発展）

新しいメンバ関数 `ShowStatus()` を追加し、
「商品名・価格・在庫」を1行で表示せよ。

☒ 教員用解説・ポイント

問題	ねらい	指導ポイント	模範コード抜粋
Q1	getter/setter の型と戻り値の対応	`void SetXxx(T)` と `T GetXxx()` の対称性を意識	`void SetItemName(string n){itemName=n;}`
Q2	setter での値検証	if文で範囲制限。固定値ではなく変数を渡す形を見せる	`if(value<0)value=0;`
Q3	状態の整合性保持	在庫は負数で意味を持たない → 0で丸める	`if(value<0)value=0;`
Q4	実行結果の確認	出力順に注目。「在庫がない！」が出るまでSell()	`m.Sell();m.Sell();m.Sell();`
Q5	状態表示	デバッグ・確認に便利。getterの活用を実感させる	`cout<<"商品:"<<itemName<<" 在庫:"<<stock<<endl;`

☒ チェックリスト

- [ ]
- [ ] setterで値をチェックできているか
- [ ]
- [ ] private変数を直接触っていないか
- [ ]
- [ ] 出力が想定通りか
- [ ]
- [ ] コードに不正アクセス部分（コメントで残してある）が理解できているか

☒ 発展予告（次回）

- Playerクラスを導入し、`SellTo(Player&)` で取引処理
- has-a関係・参照渡しの導入
- 複数商品対応（vector）へ展開

☒ まとめコメント例（授業締め）

> 「今日の目的は“商人の金庫に鍵をかける”でした。
> 直接代入では壊れる。関数を通せば守れる。
> getter/setter は、プログラム世界のルールブックです。」

Revision #2
★Created 7 November 2025 01:05:22 by youe2
Updated 9 June 2026 00:17:11 by youe2