PPA Assignment 7

Wonjoon Seol, Computer Science with Intelligent Systems, K1631098

November 26, 2016

1 Introduction

In this assignment we simulate pirate treasure hunting activities. The user will suggest which Island pirate should travel to find the treasure and the pirate, if the given island exists, will sail to the given island and dig multiple locations in the island to find the hidden treasure. In this assignment, I need to demonstrate my ability to be able to use Arrays and use static fields where necessary. (Topics from week 1,...,8).

2 Pseudocode

2.1 Class GoldCoin

Pseudocode 1: This class models a gold coin.

```
1 Initialise private String $coinNumber$
2 Initialise private static integer $totalCoin$
3 Define GoldCoin
4     Set $coinNumber$ to be totalCoin + 1
5 Define getCoinNumber
6     Return $coinNumber$
```

2.2 Class TreasureChest

Pseudocode 2: This class models a treasure chest with gold coins inside.

```
Initialise private Int $noOfGoldCoins$
Initialise private GoldCoin[] $goldCoin$
Define TreasureChest
    Set goldCoin to be goldCoin array GoldCoin of size 17
    for int i = 0 to 16 do
        Initialise new object GoldCoin at goldCoin array index i
    end
Define getNoOfGoldCoins
    Return $noOfGoldCoins$
Define takeOneGoldCoin
    if noOfGoldCoins is greater than 0 then
        Subtract one from $noOfGoldCoins$
        Intialise GoldCoin coin to be value of goldCoin array location $noOfGoldCoins$
        Set array goldCoin index $noOfGoldCoins$ to be null
        Return coin
    else
        Return null
    end
```

2.3 Class Island

Pseudocode 3: This class models island with treasures buried in one of its locations.

```
Initialise private String name
Initialise private TreasureChest[] locations
Define Island
    Set name Intialise Random rnd
    Intialise int random with random integer between 0 and array length of locations
      supplied as a parameter
    Intialise new object TreasureChest at array locations index random
    Set locations
Define getName
    Return name
Define getLocation
    Return locations
Define dig
    if i is less than array length of locations then
        if value at locations is not null then
            Initialise TreasureChest chest to be value at array locations index i
            Set array locations at index i to be null
            Return chest;
        else
            Return null;
        end
    else
        Return null;
    end
```

2.4 Class Pirate

Pseudocode 4: This class models characteristics of pirate.

```
Initialise private String name
Initialise private GoldCoin[] purse
Initialise private Island[] map
Initialise private int coinIndex
Define Pirate
    Set name Set map
    Set purse to be array GoldCoin of size 100
Define totalCoins
    Return coinIndex
Define addToPurse
    Return locations
Define search
    Initialise Island island to be null
    for int i = 0 to array size of map - 1 do
        if name equals name of island stored at array map at index i then
            Set island to be island at array map at index i
        end
    end
    Return island
Define getTreasure
    Initialise boolean isCoinTaken to be false
    for int i = 0 to number of locations in island - 1 do
        Initialise TreasureChest chest chest returned from digging location index i of island
        if chest is not null then
            Initialise int maxChestCoinNumber to be number of gold coins from chest
        end
        for int j = 0 to maxChestCoinNumber - 1 do
            Initialise GoldCoin coin to be coin returned from takeOneGoldCoin method
             from object chest
            if coin is not null then
                add coin to pirate's purse Set isCoinTaken to be true
            end
        end
        Return isCoinTaken;
    end
    Return isCoinTaken;
Define speak
    Intialise Random rnd
    switch random number between 0 to 4 do
        case 0 do
            Print statement with ", arr" as a suffix
        case 1 do
            Print statement with ", shiver me timbers!" as a suffix
        case 2 do
            Print statement with ", avast!" as a suffix
        case 3 do
            Print statement with ", ahoy, matey!" as a suffix
        case 4 do
            Print statement with ", yo, ho ho!" as a suffix
        otherwise do
            Print statement
        end
    end
```

2.5 Class TreasureHunt

Pseudocode 5: This class is going to drive our program.

```
Initialise TreasureChest[] $location1$ to be array size 13
Initialise TreasureChest[] $location2$ to be array size 13
Initialise TreasureChest[] $location3$ to be array size 13
Initialise Island $portRoyal$ with "$PortRoyal$", $location1$
Initialise Island $tortuga$ with "$Tortuga$", $location2$
Initialise Island $dominica$ with "$Dominica$", $location3$
Initialise Island[] $map$ to be array size 3
Set $map$ index 0 to be $portRoyal$
Set $map$ index 1 to be $tortuga$
Set $map$ index 2 to be $dominica$
Initialise Pirate $captainChapman$ with "$Chapbeard$", $map$
Initialise Scanner $in$
Initialise String $nextIsland$
do
    captainChapman asks user to input name of an island
    Set $nextIsland$ to read next input value
    if nextIsland is not equal to "stop" then
        captainChapman speaks the island he is searching for
        Initialise Island island to be the island returned from pirate searching given user
          input
        if island is null then
            captainChapman tells the user that given island does not exist
        else
            if captainChapman find treasure chest from island then
                captainChapman tells the user that he has found the chest
                captainChapman tells how many coins he has in his purse
            else
                captainChapman complains there is no treasure in the island
            end
        end
    end
while nextIsland is not equal to "stop"
Close scanner in
```

3 Class Diagram

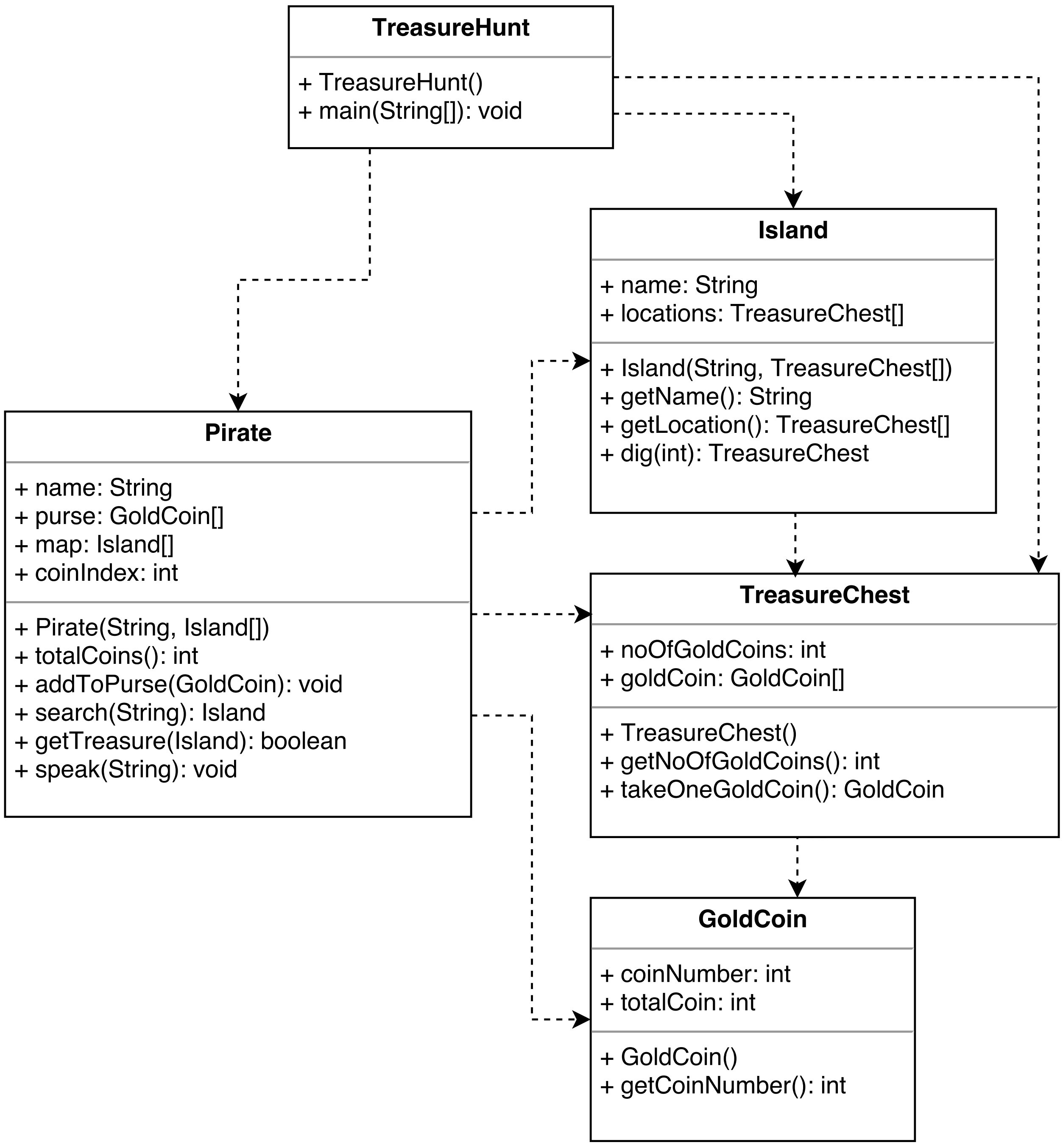

Figure 1: Treasure Hunt Class diagram.

4 Description

1. Class GoldCoin
   The class represents a gold coin. The coin has a unique coin number which is determined when it is created. Uniqueness is achieved by static field field, which keeps track of total number of objects created. Private final could have been used but same functionality is achieved by not making any setters to modify the current coin number instead. This is to comply with the style of our lecturer. totalCoin is increased as a prefix instead of post fix to make the coin number starting from 1.

2. Class TreasureChest
   The class TreasureChest generates 17 new gold coins when initialised and stores them type GoldCoin array called *goldCoin*. The method takeOneGoldCoin checks whether the gold is left in the chest and returns one gold coin after removing it from the array.

3. Class Island
   This class generates random number when initialised to store a treasure chest inside *locations* array at that random position. There are 13 locations and the treasure would be placed at one of these locations randomly. The method dig checks whether the supplied index i is not out of the bounds of the array and then checks to find whether the array at supplied index holds a treasure chest. If a treasure chest exists it removes from the array and return this chest.

4. Class Pirate
   The method *speak* is static here as they all share the same language. This is also reinforced by the fact that nothing in the speak method relies on a pirate's state. However, if the concept here is pirate words not pirate language, then I would change the speak method to non static method and alter the structure in the following way:

   - The speak method will accept text file or String value representing each pirate's unique suffixes
   - This text file or string value will be broken down and stored into elements in Arraylist by utilising String.split() or delimiters
   - A random number between 0 to the size of the arraylist - 1 will be generated
   - For every print statement it will look up element at the index 'the random number' generated above and the pirate will concatenate provided statement with its suffix. The random value may also determine whether the given phrase is positioned as prefix or post fix.

   In this way, every pirate will have different habbits/speaking style. The method getTreasure makes the pirate to dig every location and if a chest is discovered the pirate would start to take coins from the chest into his purse. An extra step is taken to ensure the coin returned is not null. A private integer field *coinIndex* is used to track number of coins and array *purse* next index value to put the coin inside. If ArrayList can be utilised then this field will not be required. Finally the method *search* would iterate every elements in the map to find the matching island. I could have used for each loop or iterator would worked here nicely too.

5. Class TreasureHunt
   This is our driver class, after watching the Dr. Martin Chapman's help and tips video guide for this assignment, I tried to make the console output as similar to his as possible. Inside my do-while loop, I put another if statement to skip rest of the codes when the user typed 'stop' inside. I didn't want the pirate to complain there was no island called stop. Integer.toString method was used to convert integer value *totalCoin* into String data type.