Tutorial-5

Object Class

Balloon buster game is very popular in carnivals across the globe and some of the organizers are now looking for a software implementation of this game. You have been approached by these organizers to implement this software. The details of this game are explained below.

The game consists of a 10x10 board with a balloon pinned at each index in this board. Some of these balloons are associated (tagged) with a prize to be won. The organizer is miser and has decided that only 50% of these total balloons will be tagged with a prize (note that two prizes can never be tagged with one balloon). He randomly selects 50% of these balloons on the board that will be tagged with a prize object. The organizer treats each type of prize objects equally.

Prize objects available with the organizers are of three categories and each prize has some weight (mentioned in grams within brackets below) associated with it as follows:

1. Soft toys: Dog (50), Cat (40), Rabbit (20)
2. Candy bars: Kit-Kat (10), Snickers (5), Five-Star (7)
3. Stationery items: Pen (2), Pencil (1), Eraser (3)

Game Rules:

1. When a player starts playing the game, he will first create a list of any 5 (unique) objects of his choice from the above mentioned prize types. Organizer guarantees that his hidden prize list includes each type of prizes mentioned above.
2. Assume that each index on the board is always pinned with a balloon (although only 50% of these will have prizes). The player never misses a shot and always hits a balloon. As it is a virtual game, you can also assume that hitting a balloon merely symbolize the balloon being hit but not actual bursting of the balloon so that the organizer has to pin a new balloon at that index.
3. Each player gets 10 chances to shoot at the balloon board. Note that each of these shots should be targeted at a new position i.e the player cannot repeat his/her previous shots in his 10 attempts.
4. After hitting a balloon, the player has to guess the prize hidden behind the board for this balloon and then give that object from his list (see bullet 1 above) to the organizer. E.g., after hitting a balloon if the player guesses that the hidden prize object is a Cat, he will retrieve the Cat from his list and then hand over this Cat to the Organizer. The Organizer will then compare this object with the one tagged with that balloon behind the board. If it is an exact match, the Organizer will give an identical copy of the tagged prize object back to the player. Player maintains a separate list of prizes he wins in the game. Every time he wins a prize he will add that prize in this list.
5. Apart from this, the player also gets some points (added or deleted) when he hits a balloon. At the start of game, player doesn't own any points. There are three scenarios for updating total points owned by the player at any hitting of balloon (all calculations are integer based):

a. Player hits an untagged balloon: total points of the player gets halved.
b. Player hits a tagged balloon and guesses the prize correctly: a soft toy would double the player's points and also add 10 extra points. A candy bar would add 20 extra points and a stationary item first increases the total points by 10% and then add 5 extra points.
c. Player hits a tagged balloon but prize guessed is incorrect: if the player guessed for prize object of type X, points equal to the absolute difference between the weight of the tagged object and the guessed object will be deducted from player's total points. Note that the total points of a player never goes below zero at any moment.

Given this game specifications, you need to simulate the whole process and at the end of the player's 10 shots you need to print the prizes in the same order that he/she won and the number of points accumulated. Take a look at the sample test case provided at the end of this description. You must use object oriented programming approach for your implementation. Specifically, you must use the concepts of inheritance, polymorphism, object comparison and object cloning taught in the lectures.

Bonus Implementation (Optional)

There is a problem with the current setup that the organizer is not aware of. Since the replenished objects are placed at the same place as they were before, the player now knows which prize is behind that balloon position. So after completing his 10 shots, the player can decide to play another round of this game. This time he knows the exact positions of all the prizes that he/she won during the last round. You being a sober person, discuss this with the organizer and he suggested the following changes to the game:

1. Whenever a player returns to play another round he/she does not create the initial list of objects he wants to win (bullet 1 in game rules above). Note that multiple rounds are played in the same program invocation.
2. The points calculation system changes in this case. Each soft toy now is valued at flat 30 points, candy bars are valued at flat 20 points and stationery is valued at flat 10 points regardless of their weights.
3. The player must only aim for those balloons that fetched him prizes during the previous round (same index on board). However, now he may not hit in the exact same order. This is done (explained below) so that he can achieve maximum possible points.
4. From his previous round, he has a list of prizes won in that round (bullet 4 in game rules above). Let that list be "S" where each entry in that list is now updated with the new points system for each object (see bullet 2 above). The player starts his game again with zero points.
5. When the player plays the i^{th} item in list S, his total points is increased by $S * S_{i-1} * S_{i+1}$. Note that for i = 0, $S_{i-1} = 1$ and for i = number of prizes won last round - 1, $S_{i+1} = 1$ for boundary conditions. Note that player wins the i^{th} item in this round, it is removed from the sequence S.
6. The player can win this round only if he is able to score maximum number of points ever possible from that sequence.

You need to come up with a strategy for the players which if they follow will get them to maximize their points. The strategy here means the order in which one should plays the i^{th} item in the sequence S. You should print this strategy along with the number of points that the player will get using that strategy. Note that your solution must be efficient and should be able to scale to larger sequences as well.
If you are attempting the bonus portion then your implementation must also work for the non-bonus portion as well.

Sample Test Case (non-bonus portion) - Blue: Input, Red: Output

```
2          <Number of players>
Player1 Player1          <Player's names space separated>
--Player1 started playing--
dog  cat  kitkat  snickers
pen Choose a coordinate
3 4
Sorry, no prize here
0 points
Choose a coordinate
4 4
Guess the prize
0
You won dog
10 points
Choose a coordinate
1 2
Sorry, no prize here
5 points
Choose a coordinate
5 5
Guess the prize
4
Sorry, you guessed wrong, it was snickers
2 points
Choose a coordinate
0 0
Sorry, no prize here
1 points
Choose a coordinate
7 3
Sorry, no prize here
0 points
Choose a coordinate
3 3
Guess the prize
2
You won kitkat
```

20 points
Choose a
coordinate 9 9
Sorry, no prize
here 10 points
Choose a
coordinate 1 9
Guess the
prize 4
You won
pen 16
points
Choose a
coordinate 8 8
Sorry, no prize
here 8 points
--Summary of Player1--
You have won dog, kitkat, pen. Total points won = 8
--Player1 started
playing-- Maximum
Points = 6330 Strategy
= kitkat, pen, dog