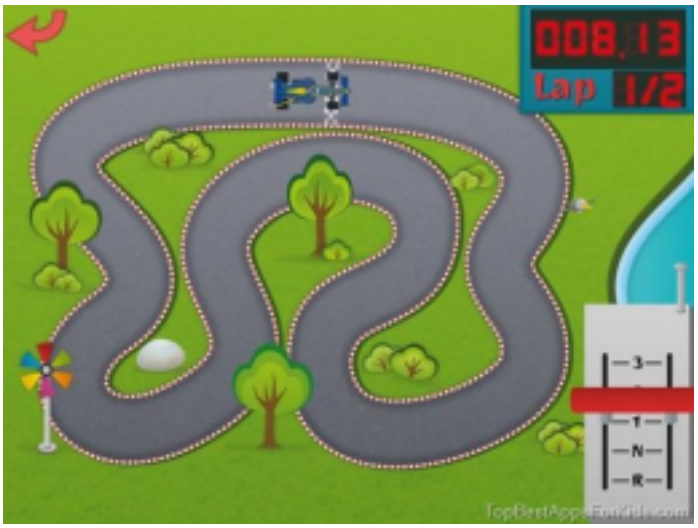
CSE201 Advanced Programming



We would like to keep this documentation really simple and short in this assignment. There is a sample test case execution shown at the end of this document that you can understand easily to code this racing game.

Like I said, this is a racing game, and like any other racing game there are obstacles and freebies on your race track. For implementing this game **you must use inheritance, polymorphism, exception handling, and define your own exceptions**. We have given the names of those exceptions as well in the sample test case.
The game starts with the user entering the total number of tiles on the race track. Each tile represents one position in one dimension. A tile can be of type **Snake, Vulture, Cricket, White, or Trampoline**. This is a single player game and once the game starts, its the computer who plays the game without any user intervention, and continues playing until the player wins.

Total number of Snakes, Vultures, Cricket, and Trampoline are randomly generated in some realistic way (ie. every tile should not have these items). You can decide randomly for each of them how many such tiles could be in your race track. All of the remaining tiles in the track are White tiles.

The player has to shake the tile he reaches to see what is hidden inside: *Snake, Vulture, Cricket, White, or Trampoline*. Upon shaking, the tile will make a specific sound, announcing which type of tile it is, and would also describe what it is going to do to the player (either make player move backward or forward). Each Snake, Vulture, Cricket, and Trampoline will have a fixed number (again randomly generated realistic number, x, where 1<= x <= 10, assuming total tiles >> 10) associated with it. For e.g., if the number associated with Snake is 7, then upon shaking a snake tile, first the snake will describe what it is going to do *(see test case)* and then also throw a custom exception **SnakeBiteException** to continue the game. Likewise, Cricket will throw

CricketBiteException, Vulture will throw **VultureBiteException**, and Trampoline will throw **TrampolineException**. Only difference is that the Trampoline would let you advance by a fixed number of Tiles (again randomly generated realistic number, x, where 1<= x <= 10, assuming total tiles >> 10) whereas Snake, Vulture and Cricket will cause player to go back a certain number of tiles (again randomly generated realistic number, x, where 1<= x <= 10, assuming total tiles >> 10). A white tile does not do anything special when shaken, and simply describes that its a white tile and the player is allowed to rest on it without any danger or freebies.

Few other simple rules of the game:

a) Computer decides how many tiles the player can move by throwing a 6 faced dice. b) Player can move out of start position (Tile-1) only if it gets 6 from the dice roll. After getting a 6 at Tile-1, the player will get another dice roll. *(see test case)*

c) The game will continue endlessly, unless the player will reach the last tile in the racing track. This event will also throw another custom exception **GameWinnerException**. Computer will also print the total moves required for winning.

d) You must use proper exception handling at all appropriate places (remember defensive programming!)

==========TESTCASE==========

>>Enter total number of tiles on the race track (length)
100
>>Setting up the race track…
>>Danger: There are X, Y, Z numbers of Snakes, Cricket, and Vultures respectively on your track! // X, Y, Z are randomly generated as described in the documentation above
>>Danger: Each Snake, Cricket, and Vultures can throw you back by A, B, C number of Tiles respectively! // A, B, C are randomly generated as described in the documentation above
>>Good News: There are T number of Trampolines on your track! // T is randomly generated as described in the documentation above
>>Good News: Each Trampoline can help you advance by D number of Tiles // D is randomly generated as described in the documentation above
>>Enter the Player Name
Josh
>>Starting the game with Josh at Tile-1
>>Control transferred to Computer for rolling the Dice for Josh
>>Hit enter to start the game
>>Game Started ==========>
>>[Roll-1]: Josh rolled 5 at Tile-1, OOPs you need 6 to start
>>[Roll-2]: Josh rolled 6 at Tile-1. You are out of the cage! You get a free roll
>>[Roll-3]: Josh rolled 5 at Tile-1, landed on Tile 6.
>> Trying to shake the Tile-6
>> Hiss…! I am a Snake, you go back 7 tiles! // assumingA=7. SnakeBiteException thrown >> Josh moved to Tile 1 as it can't go back further
>>[Roll-4]: Josh rolled 5 at Tile-1, OOPs you need 6 to start
>>[Roll-5]: Josh rolled 6 at Tile-1. You are out of the cage! You get a free roll
>>[Roll-6]: Josh rolled 3 at Tile-1, landed on Tile 4.
>> Trying to shake the Tile-4
>> I am a White tile!
>> Josh moved to Tile-4
>>[Roll-7]: Josh rolled 6 at Tile-4, landed on Tile 10.

>> Trying to shake the Tile-10
>> Chirp…! I am a Cricket, you go back 2 tiles! // assuming B=2. CricketBiteException thrown >> Josh moved to Tile 8
>>[Roll-8]: Josh rolled 6 at Tile-8, landed on Tile 14.
>> Trying to shake the Tile-14
>> Yapping...! I am a Vulture, you go back 1 tiles! // assuming C=1. VultureBiteException thrown >> Josh moved to Tile-13
>>[Roll-9]: Josh rolled 5 at Tile-13, landed on Tile 18.
>> Trying to shake the Tile-18
>> PingPong! I am a Trampoline, you advance 10 tiles//assuming D=10.TrampolineException thrown >> Josh moved to Tile-28
……
……
>>[Roll-100]: Josh rolled 5 at Tile-99, landed on Tile-99.
>>[Roll-101]: Josh rolled 4 at Tile-99, landed on Tile-99.
>>[Roll-102]: Josh rolled 1 at Tile-99, landed on Tile-100. // GameWinnerException thrown >> Josh wins the race in 102 (<Should be updated appropriately>) rolls! >> Total Snake Bites = <Should be updated appropriately>
>> Total Vulture Bites = <Should be updated appropriately>
>> Total Cricket Bites = <Should be updated appropriately>
>> Total Trampolines = <Should be updated appropriately>