TEMPLATE METHOD PATTERN

Create a skeletal class with methods that are common between algorithms.

Create a subclass for each algorithm and override the common methods from the skeletal class.

Disadvantages:

no runtime flexibility

STRATEGY PTTERN

The basic idea is to delegate tasks to encapsulated algorithms which are interchangable at runtime.

The obvious thing is was we had a lot of duplicated code in the main class. First thing I did was to look to apply the Template method pattern.

There were common things between each while loop for the four different strategies of play.The algorith of guessing the

number was defined as follow: three main parts - init_limits, do_play and compare. So I created a template method called

'play' where the steps were defined:

```
#Template method
 def play
  num = init_limits(@lower, @upper)
  @num_attempts+=1
  do_play(@lower, num, @upper)
  compare
 end
```

After that I started to derived the four different strategies of guessing the number in separate Objects. Where only the two methods were

overriden :

```
  def init_limits(lower, upper)

    raise "Abstract method called"

  end


  def do_play(lower, num, upper)

    raise "Abstract method called"

  end
```

The 'compare' method was implemented in the Participant class:

```
  def compare

    if (@num_attempts <= @max_num_attempts)

      :success

    else

      fail

    end

  end
```

as it was the same for all four startegies.

So the 'compare' along with the

```
  def reset

    @num_attempts = 0
  end


  private

  def fail

    :fail

  end
```

become 'hook'methods where the concreate implenation classes didn't know about them, but provides the flexability if we need to

override them in future.

I called the template method 'play'. Creating a 'Interface' 'PlayGame' which takes as parametter a Participant

instance and has a 'play' method which delegates the play to the participant. The Idea behind was that we can play a game

without knowing who is playing it and what startegy he follows. So 'Participant' becomes my Startegy pattern with four different

implemantaions provided in the derived classes 'paly' method was my 'Template Method Pattern. Th PayGame doesn't care how the game

is played and who is playing the game, all what is care is to play the game. The player passed in decides how to plays. So the game can be played

in four different ways based on the player passed in.