Class Design Report

```
 __  __                 _   _             
|  \/  |_   _ _ __ __ _| |_(_)_ __   __ _ 
| |\/| | | | | '__/ _` | __| | '_ \ / _` |
| |  | | |_| | | | (_| | |_| | | | | (_| |
|_|  |_|\__,_|_|  \__,_|\__|_|_| |_|\__,_|
```

(The Restaurant Game)

BY
TAIMO PEELO
MICHAEL NGUGI MICHUKI

University of Tartu

Systems Modelling Course

Fall 2014

```
                             _
 /\_|__|_._o|_   _|_ _  _ // \._   _ .__._|_o _ ._  _
/--\|_ |_| ||_)|_||_(/__>/ \_/|_)(/_|(_| |_|(_)| |_>
                              |
```

Removed classes (since HW2):

TrainingCourse

ClientStatistics

MealSatisfaction

Added classes (since HW2):

SimulationGenerator (data generation for Client's, Employees, combinatorics):

```
Client +rndClient()
Set<Client> +rndPopulation(Integer populationSize)
Employee +rndEmployee(EmployeeType employeeType)
Set<Employee> +rndStaff(Map<EmployeeType, Integer> employeeCounts)
Set<Integer> +rndCombination(Integer k, Integer n)
Set<T> +rndCombination(Integer k, Set<T> things)
String +rndPhoneNumber()
String +rndForeName()
String +rndSurname()

Set<Dish> +rndTestDishes()
Set<Beverage> +rndTestBeverages()
```

GameController (user input / simulation loop), operations nesting in call order:

```
+showLogo()
+gameLoop()

    +userOpExit()

    +mainUserMenuOpHighScoreList()

    +mainUserMenuOpPlay()

        + askPlayerInfo()
        + askRestaurantInfo(Player player)
        + createStaff()
        + askRestaurantMenu(Game game)

            +askDishInfo(int order)
            +askBeverageInfo(int order)
            +askMenuItemInfo(MenuItem item, int order)
        + dailyLoop(Game game)
```

+ showBankruptcy()

+ askRestaurantMenu(Game game)

+ dailyCommandInput(Game game)

+ getTrainableStaff(Game game)

+ assignWaiterTablesInput(Game game)

+ trainInput(Game game)

+ daySimulation(Game game)

Added attributes/fields (since HW2):

Game: (removed were start(), exit(), getHighScoreList())

+ day: Integer

+ getDay(): int

+ getRestaurant(): Restaurant

+ getClientPopulation(): Set<Client>

+ getScore(): Integer

Employee:

+ getServicedTables(): Set<Table>

+ setExperience(ExperienceLevel level)

+ baseLineClientSatisfactionPercentage() : int

+ getTrainingCost() : int

+ getSalary(): int

+ isWaiter() : boolean

+ experienceString(): String

+ titleString(): String

MenuItem:

+ getIngredientCost(): Integer

Restaurant:

+ (get/set) Name()/Address()/Menu()/City()/Staff()/Tables()

+ sellMenuItem(MenuItem item)

+ getDebtToSuppliers()

+ payDebtToSuppliers()

+ paySalaries()

+ payMonthlyCosts()

+ train(Employee employee)

+ is (High|Low|Medium)Reputation() : boolean

+ getWaiters(): List<Employee>

+ getChef() : Employee

+ getBarman() : Employee

```
  _             __
 | \ _._|_ _.  (__|_._   __|_   .__  _
 |_/(_|  |_(_|  __)|_||_|(_ |_|_||(/__>
```

JavaEnumeration:
  EmployeeType (BARMAN, CHEF, WAITER)
  ExperienceLevel (LOW, MEDIUM, HIGH)
  QualityLevel (LOW, HIGH)

List:
  List<MealOrder> : Client's Orders

Set:
  population: Set<Client>
  employees : Set<Employee>

SortedMap:
  HighScoreList = SortedMap<Game, Player> (ordering defined by Game.score)

```
 _                _
|_)_ _     _| _  /  _  _| _
| _>(/_|_|(_|(_) \_(_)(_|(/_
```

```
///////////////////////////////////
SimulationGenerator
///////////////////////////////////
  Client +rndClient():
    create Client AS (client)
     set (client) name to (SimulationGenerator.rndForeName())
     set (client) surname to (SimulationGenerator.rndSurname())
     set (client) phone number to
(SimulationGenerator.rndPhoneNumber())
     set (client) tax code to (SimulationGenerator.rndTaxCode())
    return (client)

  Set<Client> +rndPopulation(Integer populationSize):
    create Set<Client> AS (population)
    while (population) contains less than populationSize Clients
      add (SimulationGenerator.rndClient()) to (population)
     return (population)

  Employee +rndEmployee(EmployeeType employeeType):
    create Employee AS (employee)
    set (employee) name to (SimulationGenerator.rndForeName())
    set (employee) surname to (SimulationGenerator.rndSurname())
    set (employee) experience to Low
    set (employee) type to (employeeType)
    set (employee) salary to (#Employee Salary (employee))
    if (employeeType) is Chef
      set (employee) tax code to
(SimulationGenerator.rndTaxCode())
     return (employee)

  Set<Employee> +rndStaff(Map<EmployeeType, Integer>
employeeCounts):
    create Set<Employee> as (result)
    for each (employeeType, employeeCount) FROM (employeeCounts)
      create Set<Employee> as (employees)
       while (employees) contains less than (employeeCount)
elements
         add SimulationGenerator.rndEmployee(employeeType) to
```

```
(employees)
       add (employees) to (result)
     return result

  Set<Integer> +rndCombination(Integer k, Integer n):
    create empty Set<Integer> of initial allocated size k AS
(combination)
    if (k == n)
      add 1..n to (combination)
       return (combination)
     else
       while (combination) has less than k elements
         add random Integer from range (1..n) to (combination)
     return (combination)

  Set<T> +rndCombination(Integer k, Set<T> things):
    set (SimulationGenerator.rndCombination(k, length of (things))
AS (numberCombination)
    for each (numericIndex) from (combination)
      add (things) element at (numericIndex) to (result)
    return (result)

  String +rndPhoneNumber():
    get random country code from predefined international calling
country codes list AS (prefix)
    generate random 8-digit number AS (suffix)
    return concatenation of (prefix) AND single whitespace AND
(suffix)

  String +rndForeName():
    return (random element from pre-defined forenames List)

  String +rndSurname():
    return (random element from pre-defined surnames List)

  String +rndTaxCode():
    return (random 11-digit string)
///////////////////////////////////
END OF SimulationGenerator
///////////////////////////////////

///////////////////////////////////
Employee
///////////////////////////////////
```

```
  integer +getTrainingCost():
    if (employee) type is not Waiter OR Chef OR Barman
       throw Exception
     else
       if (employee) type is Waiter
         return 800
       else
         return 1200

  boolean +increaseExperience():
    if (employee) experience is not Low OR Medium
       throw Exception
     else if (employee) experience is Low
       set (employee) experience to Medium
     else if (employee) experience is Medium
       set (employee) experience to High

  integer +getSalary():
    baseSalary = 200 if (employee) is Waiter else 300
    if (employee) experience is High
      return (baseSalary) + 200
    else if (employee) experience is Medium
      return (baseSalary) + 100
    return (baseSalary)
///////////////////////////////////
END OF Employee
///////////////////////////////////

(game starts from Controller.main)

# Assign Tables(Game)
  TODO

# Create Menu
  create Set<Dish> with 5 unspecified dishes AS (dishes)
  create Set<Beverage> with 5 unspecified dishes AS (beverages)


# De-serialize High Score List
  de-serialize HighScoreList from saved file 'muratino-hs.ser'

# Display High Score List(HighScoreList)
  for each pair of (game, player) IN HighScoreList
    display rank (index of pair)
```

```
     display player name
     display game score
     display newline

# Game Day Simulation(Game)
  set (#Semi-Random Table Combination(restaurant)) AS
(occupiedTables)
  set (#Semi-Random Client Combination(restaurant)) AS
(visitingClients)
  TODO

# End Of Game Day
  display "Another business day has passed."
  display "Budget available for the next day is: " and
(restaurant.availableBudget)
  TODO

# Exit Game
  terminate game process

# Expect User Input (instructions, pattern)
  display instructions
  wait for user input AS (input)
  if (input) does not match (pattern)
    display (input) and " is not valid command, come again, chief:
"
   return (#Expect User Input (pattern))
  else
   return (input)

# Main
  population = SimulationGenerator.rndPopulation(18)
  employees = SimulationGenerator.rndStaff({Barman:1, Chef:1,
Waiter: 3})
  while (user has not chosen to exit the game)
    # Show Player Options Menu
    # Wait User Input
    # React To User Input

# Save High Score List
  serialize HighScoreList into saved file 'muratino-hs.ser'

# Semi-Random Client Combination(restaurant, population)
   numClients = if (restaurant.reputation) is >= 30 set
```

```
restaurant.tables.length*2 AS (numClients)
   else if (restaurant.reputation) is < 15 set 4 AS (numClients)
   else set 10 AS (numClients)
   return SimulationGenerator.rndCombination(numClients,
population)

# Semi-Random Table Combination(restaurant)
   if (restaurant.reputation) is >= 30
     return (1..9)
   if (restaurant.reputation) is < 15
     return (#Random Combination(2, 9))
   else
     return (#Random Combination(5, 9))

# Show Player Options Menu
  if (no Game ongoing)
     display "1. Start New Game"
      display newline
      display "2. View High Score List"
      display newline
      display "42. Exit Game"
      display newline
      (#Expect User Input ("1|2|42")) AS (cmd)
      if (cmd) is 1
        (#Start Game)
      else if (cmd) is 2
        (#Show High Score List)
      else if (cmd) is 42
        (#Exit Game)
  else if (not end of month AND restaurant.availableBudget < 0)
     display "Game has ended with a bankruptcy, funds missing" AND
(-restaurant.availableBudget) AND " EUR"
  else if (End of month has ended and restaurant.availableBudget
is < 0)
  else if ()

# Start Game
  display Ascii Art Game Logo
  display newline
  display "It is a beautiful autumn in a cold northern country of
Estonia."
  display newline
  display "Almost all pale yellow leaves are fallen from birches
and maples."
```

```
  display newline
  display "In the creeping winter students roam in their coats and
hats."
  display newline
  display "And you have decided to open up a new cozy restaurant."
  display newline
  display "What is your name, brave entrepreneur?"
  (#Expect User Input("[a-zA-z]+\n")) AS (playerName)
  display "How do you want to name your restaurant, " and
(playerName) and "?"
  (#Expect User Input("[a-zA-z]+\n")) AS (restaurantName)
  display "What city will you open your restaurant in, " and
(playerName) and "?"
  (#Expect User Input("[a-zA-z]+\n")) AS (city)
  display "What is the address where you have acquired the
restaurant space, " and (playerName) and "?"
  (#Expect User Input("[a-zA-z]+\n")) AS (restaurantAddress)
  create new Player as (player)
  set (player) name to (playerName)
  create new Restaurant as (restaurant)
  set (restaurant) name to (restaurantName)
  set (restaurant) city to (city)
  set (restaurant) address to (restaurantAddress)
  set (restaurant) owner to (player)

# Train Staff

# View Game Statistics
  Staff Names

# Show High Score List
  (#De-serialize High Score List) AS (highScoreList)
  (# Display High Score List(highScoreList))
  display "Press <Enter> to continue"
  display newline
  (#Expect User Input("\n"))
```