**Exercise 1**

**City**

-name : String

-population : int

-budget : double

+City (name: string , population : int, budget: double)

+getBudget(): double

+getPopulation(): int

Class ***City***:

* + Attributes:
    - ***name***: The name of the City.
    - ***population***: the population of the City.
    - ***budget***: The budget of the City.
  + Methods:
    - ***City(name:*** *string****, population:*** *int****, budget:*** *double****)***: Constructor.
    - ***getBudget( )***: This method returns the budget of the city.
    - ***getPopulation( )***: This method returns the population of the city.

|  |
| --- |
| **QUESTION**: Translate into Java code the class ***City*** (without getters). |

**Answer:**

public class City { ……………… /6

private String name;

private int population;

private double budget;

public City(String s, int p, double b) { ………………/3

name = s;

population = p;

budget = b;

}

public City(City c) { ……………… /3

name = c.name; ……………… 1

population = c.population; ……………… 1

budget = c.budget; ……………… 1

}

public int getPopulation() {

return population;

}

public double getBudget() {

return budget;

}

}

**Exercise 2:**

Let’s consider the same class ***City*** described in exercise 1.

\* -arrCity 1

**City**

-name : String

-population : int

-budget: double

+City (name: string , population : int,

budget: double)

+getPopulation(): int

+getBudget(): double

**Country**

-name: String

-continent : String

+Country (name: string, continent: string, size: int)

+getContinent(): String

+addCity(c: City): boolean

+splitCitiesOnBudget(high:City[], low:City[],b: int)

+calculatePopulation(b: double): int

Class ***Country:***

* + Attributes:
    - ***name***: Name of the country.
    - ***continent***: Name of the continent where the country is located (eg. Asia, Africa, Europe etc.).
  + Methods:
    - ***Country(name:*** *string****, continent:*** *string****, size:*** *int****)***: Constructor
    - ***getContinent***(): This method returns the name of the continent where the country is located.
    - ***addCity(c:*** *City****)***: this method adds the city ***c*** to the country. It returns ***true*** if the city ***c*** is inserted. Otherwise, it returns ***false***;
    - ***splitCitiesOnBudget(high:****City[]****, low:****City[]****, b:*** *int****)***: This method splits the cities based on the budget ***b***. Cities with budget higher than ***b*** are added to ***high***, others are added to ***low***.
    - ***calculatePopulation(b:*** *double****)***: This method returns the total population living in cities with budget less than b.

|  |
| --- |
| **QUESTION**: Translate into Java code the class **Country** (without getters). |

**Answer:**

public class Country { ……………… /21

private String name;

private String continent;

private City[] arrCity; ……………… 1

private int nbCities; ……………… 1

public Country(String s, String c, int size) { ……………… /2

name = s;

continent = c;

arrCity = new City[size]; ……………… 1

nbCities = 0; ……………… 1

}

public String getContinent() {

return continent;

}

public boolean addCity(City c) { ……………… /5

if (nbCities < arrCity.length) { ……………… 1

arrCity[nbCities] = new City(c); ……………… 1 + 1

nbCities++; ……………… 1

return true; ……………… 0.5

}

else

return false; ……………… 0.5

}

public void splitCities(City high[], City low[], double b) { ……………… /7

int j=0, k=0; ……………… 1

for (int i = 0; i < nbCities; i++) { ……………… 1

if (arrCity[i].getBudget() >= b) { ……………… 1

high[j] = arrCity[i]; ……………… 1

j++; ……………… 1

}

else {

low[k] = arrCity[i]; ……………… 1

k++; ……………… 1

}

}

}

public int calculatePopulation(double b) { ……………… /5

int n = 0; ……………… 1

for (int i = 0; i < nbCities; i++) { ……………… 1

if (arrCity[i].getBudget() < b ) ……………… 1

n += arrCity[i].getPopulation(); ……………… 1

}

return n; ……………… 1

}

}

**Exercise 3:**

Let’s consider the same class ***Country*** described in exercise 2.

**UnitedNations**

-address : String

+UnitedNations (address: string)

+getAsianCountries():Country[]

+poorCountry(c: String, b:double): Country

1 -arrCountry 200

**Country**

-name: String

-continent : String

+Country (name: string, continent: string, size: int)

+getContinent(): String

+addCity(c: City): boolean

+splitCitiesOnBudget(high:City[], low:City[],b: int)

+calculatePopulation(b: double): int

Class ***UnitedNations***:

* + Attributes:
    - ***address***: The address of the United Nations head office.
  + Methods:
    - ***UnitedNations (address:*** *string****)***: Constructor
    - ***getAsianCountries***(): This method returns an array containing all countries located in Asia.
    - ***poorCountry(c:*** *String****, b:*** *double****)***: This method returns the country located in continent ***c*** having the largest population living in cities with budget less than b.

|  |
| --- |
| **QUESTION**: Translate into Java code the class **UnitedNations*.*** |

**Answer:**

public class UnitedNations { ………………/18

private String address;

private Country arrCountry[]; ……………… 1

private int nbCountries; ……………… 1

public UnitedNations(String adr) { ………………/2

address = adr;

arrCountry = new Country[200]; ……………… 1

nbCountries = 0; ……………… 1

}

public Country[] getAsianCountries() { ………………/7

Country result[] = new Country[nbCountries]; ……………… 1

int j = 0; ……………… 1

for (int i = 0; i < nbCountries; i++) { ……………… 1

if (arrCountry[i].getContinent().equals("Asia")) { ………… 1

result[j] = arrCountry[i]; ……………… 1

j++; ……………… 1

}

}

return result; ……………… 1

}

public Country poorCountry(String c, double b) { ………………/7

Country poor = null; ……………… 1

for (int i = 0; i < nbCountries; i++) { ……………… 1

if ( ( arrCountry[i].getContinent().equals(c) ) && ……… 1

( poor == null || ……1

arrCountry[i].calculatePopulation(b) > …………… 1

poor.calculatePopulation(b) ) )

poor = arrCountry[i]; ……………… 1

}

return poor; ……………… 1

}

}