Manavendra Sen | 199302058 | IT 3A

OOP LAB 4

EXERCISE 4.1

You are given the skeleton code for four incomplete classes named Pokemon, Pokeball, Trainer and a driver class named Test. You have to complete the code for all the classes as per the given specifications.

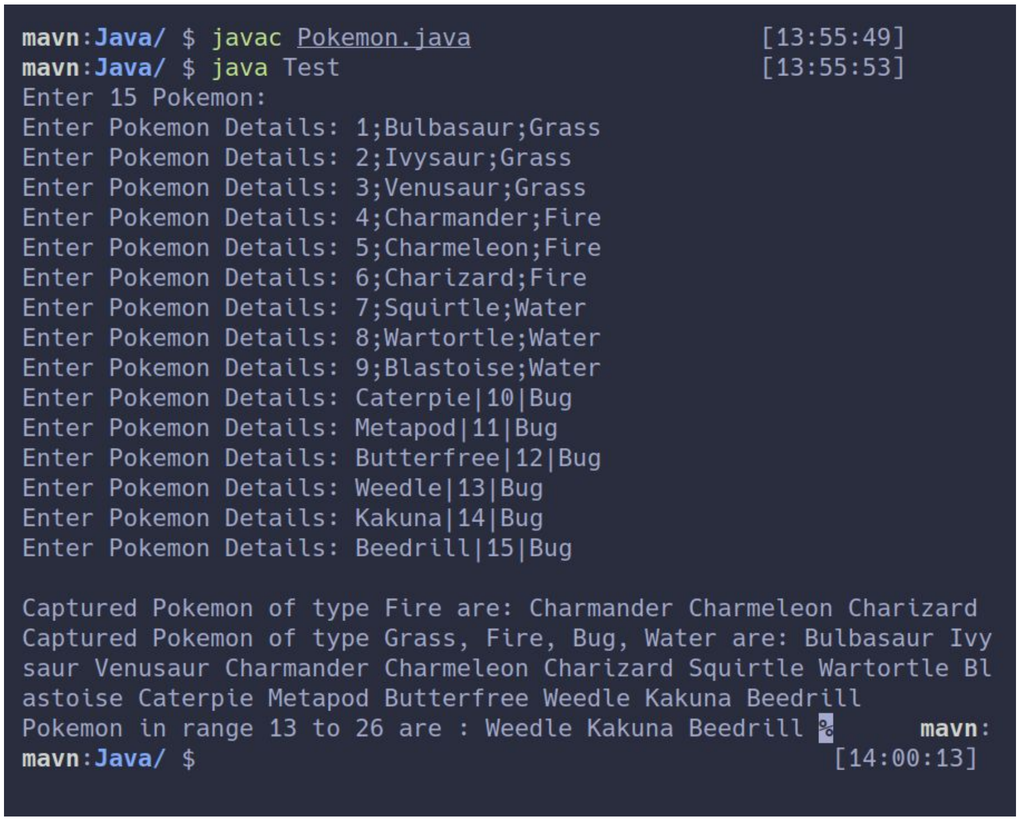
```
import java.util.*;
import java.io.*;

class Pokemon {
 private String name; // Name of the Pokemon
 private int id; // id of the Pokemon
 private String type; // type of the Pokemon

 public int getId() {
   return id;
 }

 public String getName() {
   return name;
 }

 public String getType() {
   return type;
 }

 public void getPokemon() {
   System.out.println("#" + id + "\n" + name + "\n" + type);
 }

 @Override
 public String toString() {
   return (id + " " + name + " " + type);
 }
 Pokemon(String pokemon) {
   // Assuming a correct parameter has been passed.
   StringTokenizer st = new StringTokenizer(pokemon, ";|");
   if (pokemon.indexOf("|") == -1) {
     this.id = Integer.parseInt(st.nextToken());
     this.name = st.nextToken();
     this.type = st.nextToken();
   } else {
     this.name = st.nextToken();
     this.id = Integer.parseInt(st.nextToken());
     this.type = st.nextToken();
   }
 }
} //

class Pokeball {
 private Pokemon pokemon;

 public Pokemon getPokemon() {
   return pokemon;
 }

 public void setPokemon(Pokemon pokemon) {
   this.pokemon = pokemon;
 }

 Pokeball(Pokemon pokemon) {
   this.pokemon = pokemon;
 }

 @Override
 public String toString() {
   return this.pokemon.toString();
 }
}
```

```
class Trainer {
 public static Vector<Pokeball> collection = new Vector<Pokeball>(); // list of
 // pokeballs

 public static void capturePokemon(Pokemon pokemon) {
   /*
    * This method captures a pokemon with a pokeball and adds to the trainer's
    * collection
    */
   Pokeball p = new Pokeball(pokemon);
   collection.add(p);
 }

 public static Pokemon[] getPokemonWithType(String type) {
   /*
    * This method returns all the pokemons with given type. If no such pokemon is
    * found then it returns null.
    */
   Pokemon result[] = new Pokemon[10];
   int index = 0;
   for (int i = 0; i < collection.size(); i++) {
     if (collection.get(i).getPokemon().getType().equals(type)) {
       result[index++] = collection.get(i).getPokemon();
     }
   }
   return result;
 }

 public static Pokemon[] getPokemonsWithGivenTypes(String[] types) {
   /*
    * This method returns all the pokemons whose type matches with one of the types
    * given in the array types[]. If no such pokemons is found then it returns
    * null.
    */
   Pokemon result[] = new Pokemon[20];
```

```
    int index = 0;
    for (int i = 0; i < collection.size(); i++) {
      for (int j = 0; j < types.length; j++) {
        if (collection.get(i).getPokemon().getType().equals(types[j])) {
          result[index++] = collection.get(i).getPokemon();
        }
      }
    }
    return result;
  }

  public static Pokemon[] getPokemonsInRange(int minId, int maxId) {
    /*
     * This method returns all the pokemons whose id falls between minId and maxId
     * (both parameters inclusive)
     */
    Pokemon result[] = new Pokemon[10];
    int index = 0;
    for (int i = 0; i < collection.size(); i++) {
      int id = collection.get(i).getPokemon().getId();
      if (id >= minId && id <= maxId) {
        result[index++] = collection.get(i).getPokemon();
      }
    }
    return result;
  }
}

class Test {
  public static Pokemon readPokemon() throws IOException {
    /*
     * This method reads the pokemon details and returns the Pokemon instance.
     * Values to be read from System.in are: 1. Name of Pokemon, 2. Id of Pokemon,
     * 3. Type of the Pokemon, 4. Name format (1 for pipe(|) separated and 2 for
     * semicolon separated)
     */
```

```java
  Scanner sc = new Scanner(System.in);
  System.out.print("Enter Pokemon Details: ");
  String input1 = sc.next();
  Pokemon pk = new Pokemon(input1);
  return pk;
} // End of readPokemon() Method

public static void main(String args[]) throws IOException {
  /*
   * 1. Write java code for reading details of 15pokemons and add them in the
   * static list of Trainer class.
   */
  System.out.print("Enter 6 Pokemon: \n");

  for (int i = 1; i <= 6; i++) {
    Trainer.capturePokemon(Test.readPokemon());
  }

  /*
   * 2. Write java code for displaying all the Pokemons with type “Fire” from
   * static list field of Trainer class
   */
  System.out.print("\nCaptured Pokemon of type Fire are: ");
  Pokemon array[] = Trainer.getPokemonWithType("Fire");
  int i = 0;
  while (array[i] != null) {
    System.out.print(array[i++].getName() + " ");
  }

  /*
   * 3. Write java code for displaying the Pokemons with types “Grass”, “Fire”,
   * “Bug”, “Water” from static list of Trainer class
   */

  System.out.print("\nCaptured Pokemon of type Grass, Fire, Bug, Water are: ");
```

```java
   String[] types = { "Grass", "Fire", "Bug", "Water" };
   Pokemon array2[] = Trainer.getPokemonsWithGivenTypes(types);
   i = 0;
   while (array2[i] != null) {
     System.out.print(array2[i++].getName() + " ");
   }
   /*
    * 4. Write java code for displaying all the pokemons whose id falls in the
    * range minId = 13 and maxId = 26 from static list of Trainer class
    */
   System.out.print("\nPokemon in range 13 to 26 are : ");
   Pokemon array3[] = Trainer.getPokemonsInRange(13, 26);
   i = 0;
   while (array3[i] != null) {
     System.out.print(array3[i++].getName() + " ");
   }
 }// End of main() Method
}// End of Test
```



```
mavn:Java/ $ javac Pokemon.java                               [13:55:49]
mavn:Java/ $ java Test                                        [13:55:53]
Enter 15 Pokemon:
Enter Pokemon Details: 1;Bulbasaur;Grass
Enter Pokemon Details: 2;Ivysaur;Grass
Enter Pokemon Details: 3;Venusaur;Grass
Enter Pokemon Details: 4;Charmander;Fire
Enter Pokemon Details: 5;Charmeleon;Fire
Enter Pokemon Details: 6;Charizard;Fire
Enter Pokemon Details: 7;Squirtle;Water
Enter Pokemon Details: 8;Wartortle;Water
Enter Pokemon Details: 9;Blastoise;Water
Enter Pokemon Details: Caterpie|10|Bug
Enter Pokemon Details: Metapod|11|Bug
Enter Pokemon Details: Butterfree|12|Bug
Enter Pokemon Details: Weedle|13|Bug
Enter Pokemon Details: Kakuna|14|Bug
Enter Pokemon Details: Beedrill|15|Bug

Captured Pokemon of type Fire are: Charmander Charmeleon Charizard
Captured Pokemon of type Grass, Fire, Bug, Water are: Bulbasaur Ivy
saur Venusaur Charmander Charmeleon Charizard Squirtle Wartortle Bl
astoise Caterpie Metapod Butterfree Weedle Kakuna Beedrill
Pokemon in range 13 to 26 are : Weedle Kakuna Beedrill %      mavn:
mavn:Java/ $                                                [14:00:13]
```

EXERCISE 3.2
A. Consider a class named ‘Address’ which encapsulates the address of any particular person having attributes as:
–line1:String
–line2:String
–line3:String
–city:char[]
–state:char[]
–pin:String

The class supplies only one parameterized constructor that receives only one parameter of type String which embeds the values of all the attributes in $ separated form as per the following format: **“line1$line2$line3$city$state$pin**”.

The special character($) is used as a separator to separate the values of line1, line2, line3,city, state and pin attributes. The class supplies accessor methods for every instance field. All accessor methods return only String type value. Implement the Address class in java as per mentioned specification.

B. Considering the availability of the code of class Address of (A) in this question, complete the implementation of the following class named ‘AddressList’ as per commented specification given.

C. Write a suitable driver class named Test for a class named ‘AddressList’ and test the behavior of all the methods.

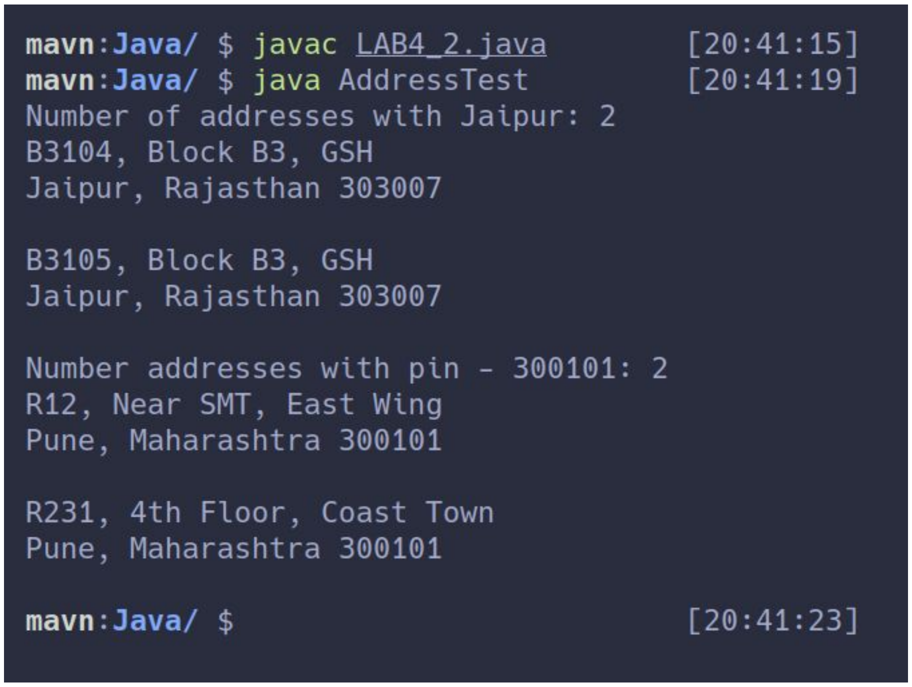
```java
import java.util.StringTokenizer;

class Address {
 private String line1;
 private String line2;
 private String line3;
 private char[] city;
 private char[] state;
 private String pin;

 // Constructor takes in a string and splits it into all the instance fields
 Address(String arg) {
   // Using string tokenizers to split the string input into separate fields
   StringTokenizer st = new StringTokenizer(arg, "$");
   this.line1 = st.nextToken();
   this.line2 = st.nextToken();
```

```
  this.line3 = st.nextToken();
  this.city = st.nextToken().toCharArray();
  this.state = st.nextToken().toCharArray();
  this.pin = st.nextToken();
}

// ACCESSORS
public String getLine1() {
  return line1;
}

public String getLine2() {
  return line2;
}

public String getLine3() {
  return line3;
}

// We wish to return a string, I made a anonymous String object to return
public String getCity() {
  return new String(city);
}

// We wish to return a string, I made a anonymous String object to return
public String getState() {
  return new String(state);
}

public String getPin() {
  return pin;
}
// Overriding the toString method to display the address in an easy manner
@Override
public String toString() {
   return (this.getLine1() + ", " + this.getLine2() + ", " + this.getLine3() + "\n" +
this.getCity() + ", " + this.getState() + " " + this.getPin());
 }
}

// AddressList Class

// Provides all class functions to perform various operation on an array of
// addresses
class AddressList {
 public static int countAddressWithCity(Address[] addressList, String city) {

   // initializing the result to 0 and incrementing it when a match is found
   int res = 0;

   // if either the arguments are null or the addressList provided is empty we
   // return -1 as specified in the question
   if (addressList == null || city == null || addressList.length == 0)
     return -1;

   // for all the addresses passed through the argument, we check if the
   // address's city is equal to the one provided
   for (Address address : addressList) {
     if (address.getCity().equals(city)) // .equals returns a boolean
       res++;
   }
   return res;
 }

 public static int countAddressWithPin(Address[] addressList, String pin) {

   // initializing the result to 0 and incrementing it when a match is found
   int res = 0;

   // if either the arguments are null or the addressList provided is empty we
   // return -1 as specified in the question int res = 0;
```

```
  if (addressList == null || pin == null || addressList.length == 0)
    return -1;

  // for all the addresses passed through the argument, we check if the
  // address's pin is starts with the one provided
  for (Address address : addressList) {
    if (address.getPin().startsWith(pin))
      res++;
  }
  return res;
}

public static Address[] getAddressWithPin(Address[] addressList, String pin)       {

  if (addressList == null || pin == null || addressList.length == 0)
    return null;

  // initialized i for using it as an index to add to the result array;
  int i = 0;

  // we call the function and add to the result array which is then returned
  Address res[] = new Address[AddressList.countAddressWithPin(addressList, pin)];

  for (Address address : addressList) {
    if (address.getPin().startsWith(pin))
      res[i++] = address;
  }
  return res;
}

public static Address[] getAddressWithCity(Address[] addressList, String city) {
  if (addressList == null || city == null || addressList.length == 0)
    return null;

  // initialized i for using it as an index to add to the result array;
  int i = 0;
```

```java
   // we call the function and add to the result array which is then returned
   Address res[] = new Address[AddressList.countAddressWithCity(addressList, city)];

   for (Address address : addressList) {
     if (address.getCity().equals(city))
       res[i++] = address;
   }
   return res;
 }
}

class AddressTest {
 public static void main(String[] args) {
   // Sample inputs
   Address a1 = new Address("12 Street$A Block$Asdf$Delhi$New Delhi$102022");
   Address a2 = new Address("R12$Near SMT$East Wing$Pune$Maharashtra$300101");
   Address a3 = new Address("R231$4th Floor$Coast Town$Pune$Maharashtra$300101");
   Address a4 = new Address("B3104$Block B3$GSH$Jaipur$Rajasthan$303007");
   Address a5 = new Address("B3105$Block B3$GSH$Jaipur$Rajasthan$303007");

   // Making an array out of the entries
   Address addresses[] = { a1, a2, a3, a4, a5 };

   // Calculating the number of addresses which have jaipur in the city field
   int a = AddressList.countAddressWithCity(addresses, "Jaipur");
   System.out.println("Number of addresses with Jaipur: " + a);

   // Getting back the details of addresses in jaipur
   Address[] b = AddressList.getAddressWithCity(addresses, "Jaipur");

   // Displaying
   for (Address address : b) {
     System.out.println(address);
     System.out.println();
   }
```

```java
    // Calculating the number of addresses which have 300101 in the pin field
    int c = AddressList.countAddressWithPin(addresses, "300101");
    System.out.println("Number addresses with pin - 300101: " + c);

    // Getting back the details of addresses in 300101 as pin
    Address[] d = AddressList.getAddressWithPin(addresses, "300101");

    // Displaying
    for (Address address : d) {
      System.out.println(address);
      System.out.println();
    }
  }
}
```



```
mavn:Java/ $ javac LAB4_2.java                  [20:41:15]
mavn:Java/ $ java AddressTest                   [20:41:19]
Number of addresses with Jaipur: 2
B3104, Block B3, GSH
Jaipur, Rajasthan 303007

B3105, Block B3, GSH
Jaipur, Rajasthan 303007

Number addresses with pin - 300101: 2
R12, Near SMT, East Wing
Pune, Maharashtra 300101

R231, 4th Floor, Coast Town
Pune, Maharashtra 300101

mavn:Java/ $                                    [20:41:23]
```