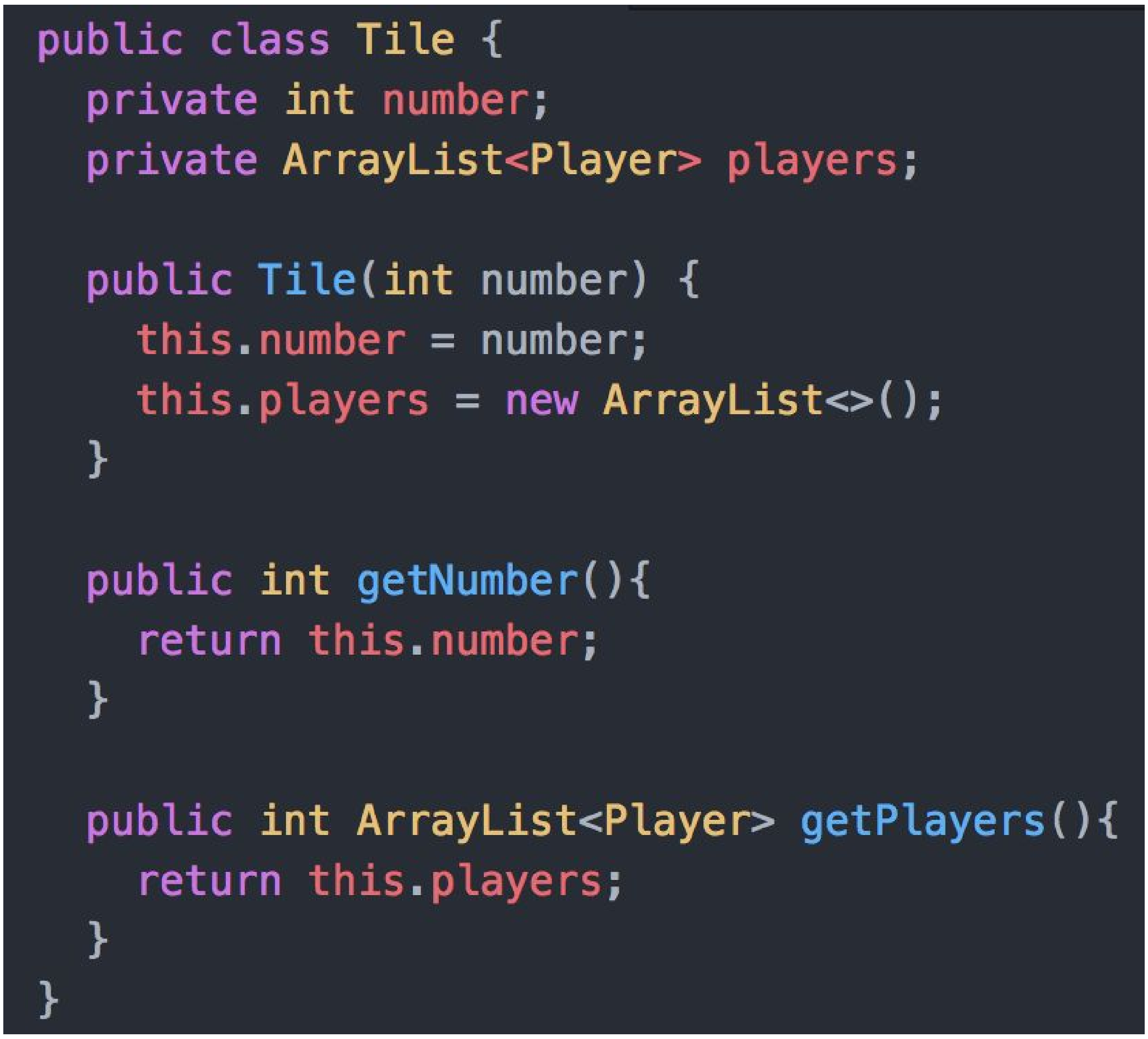
Evidence for Implementation and Testing Unit.

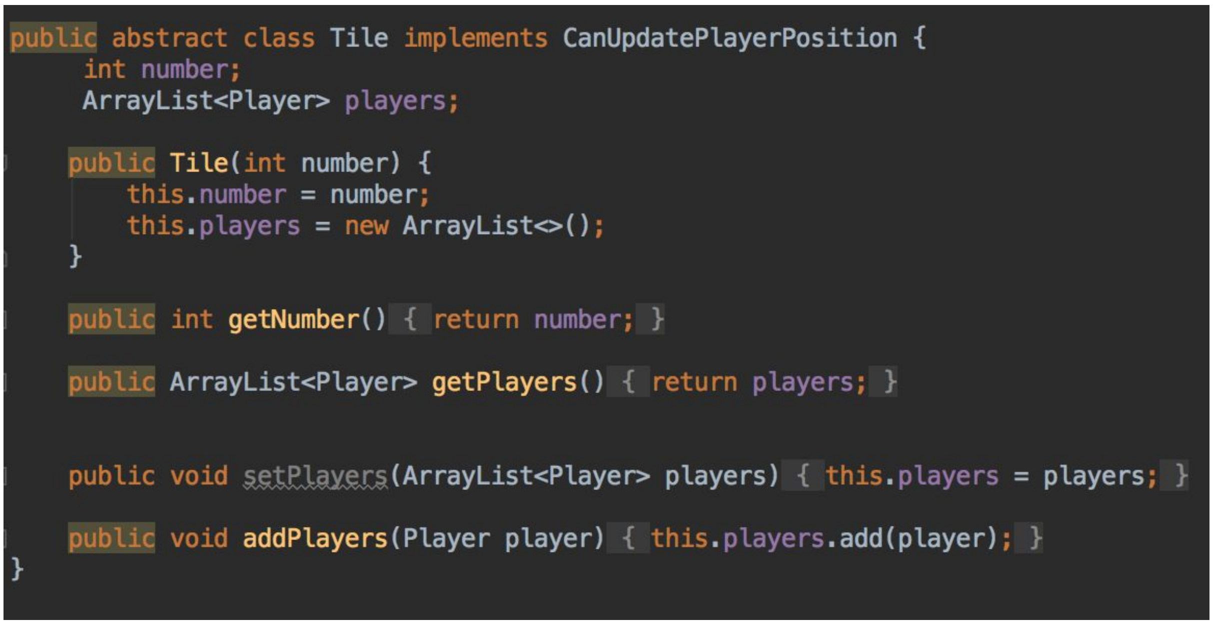
Gabriela Lewandowska
E15
18 September 2017

I.T 1- Demonstrate one example of encapsulation that you have written in a program.

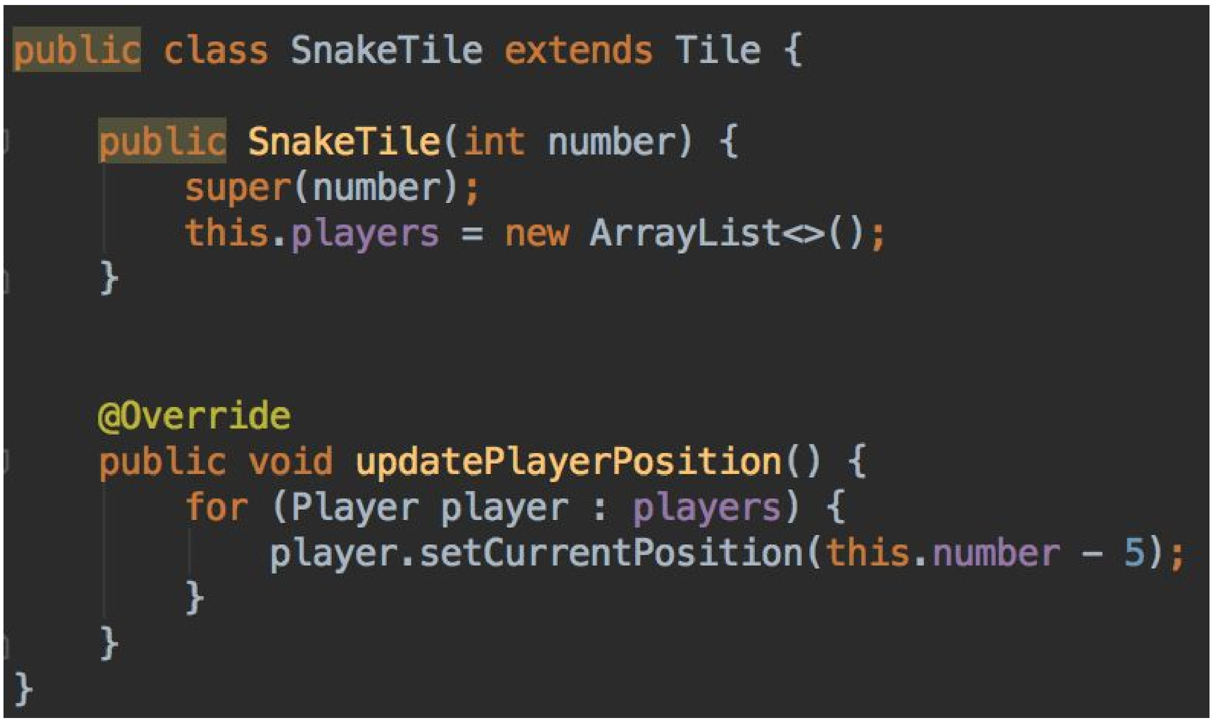


```
public class Tile {
  private int number;
  private ArrayList<Player> players;

  public Tile(int number) {
    this.number = number;
    this.players = new ArrayList<>();
  }

  public int getNumber(){
    return this.number;
  }

  public int ArrayList<Player> getPlayers(){
    return this.players;
  }
}
```

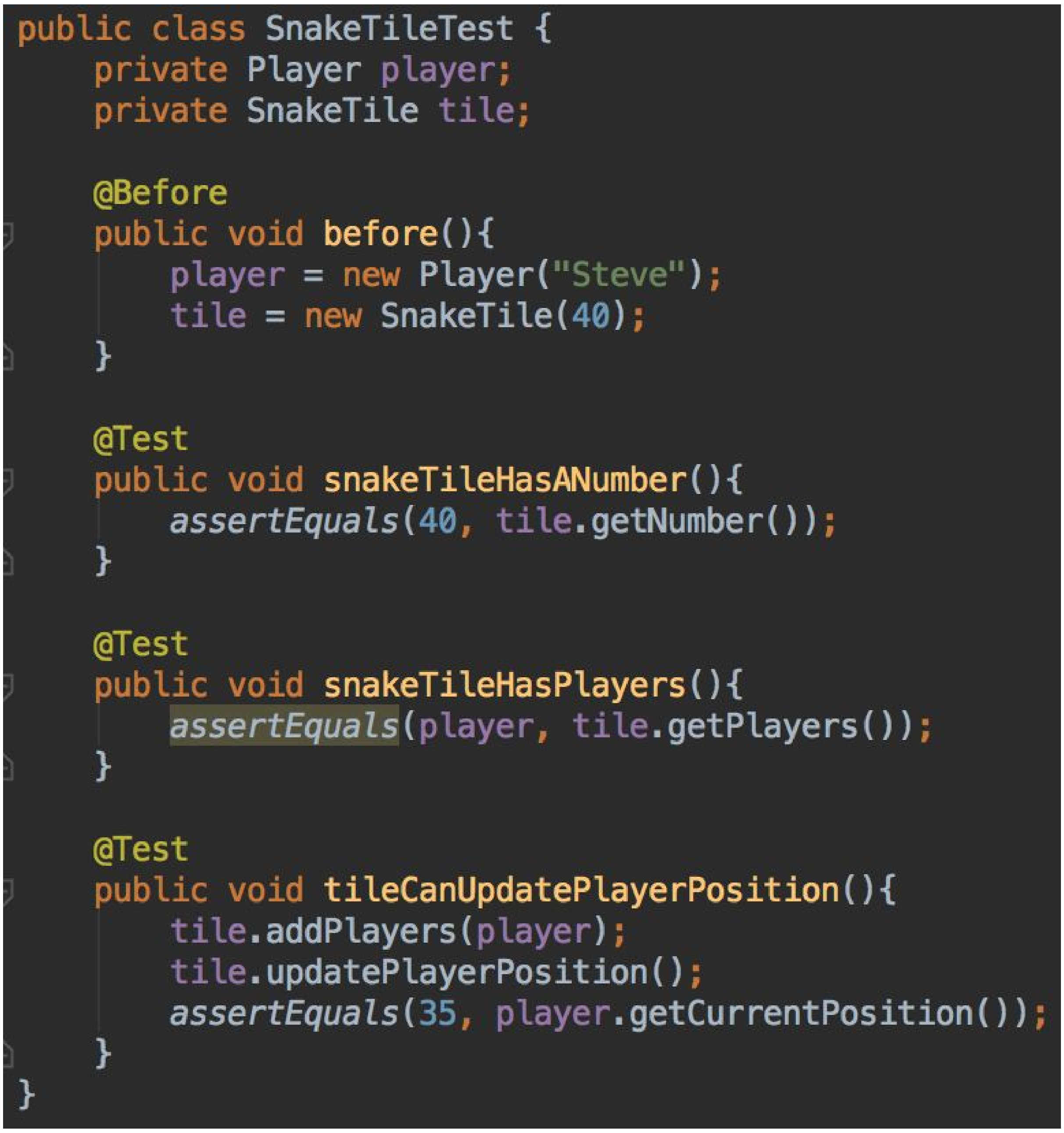
I.T 2 - Inheritance in a program.



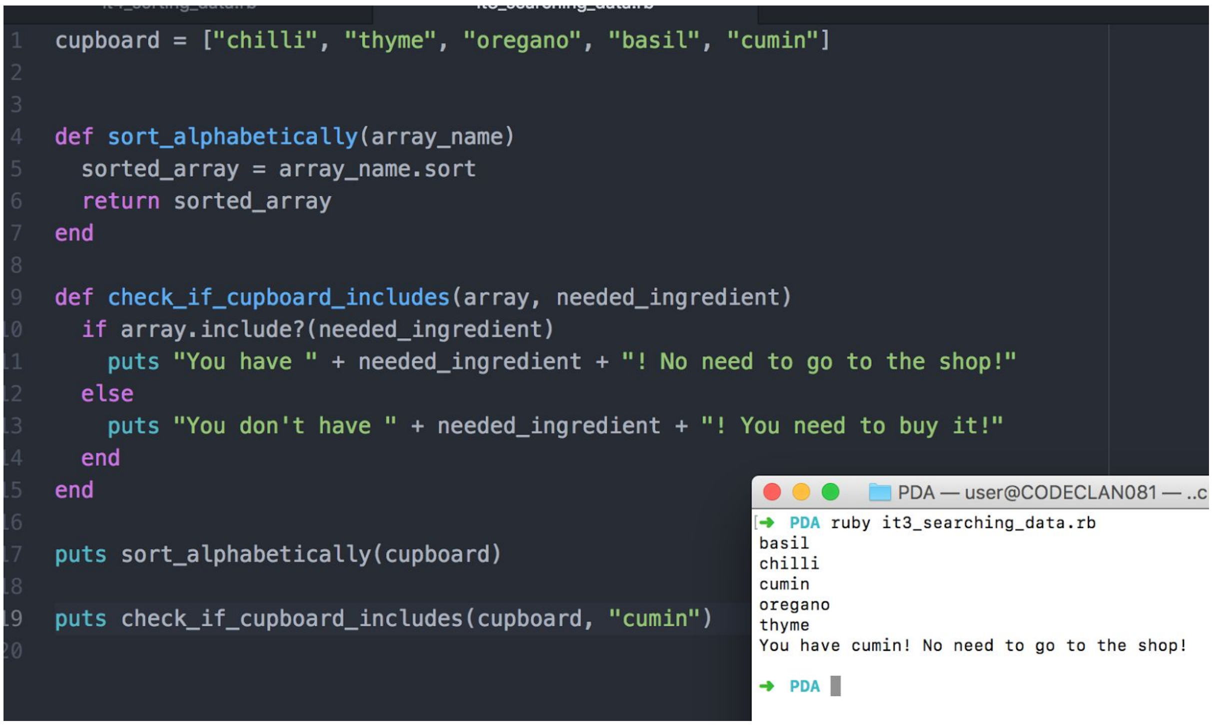
```java
public abstract class Tile implements CanUpdatePlayerPosition {
     int number;
     ArrayList<Player> players;

    public Tile(int number) {
        this.number = number;
        this.players = new ArrayList<>();
    }

    public int getNumber() { return number; }

    public ArrayList<Player> getPlayers() { return players; }


    public void setPlayers(ArrayList<Player> players) { this.players = players; }

    public void addPlayers(Player player) { this.players.add(player); }
}
```



```java
public class SnakeTile extends Tile {

    public SnakeTile(int number) {
        super(number);
        this.players = new ArrayList<>();
    }


    @Override
    public void updatePlayerPosition() {
        for (Player player : players) {
            player.setCurrentPosition(this.number - 5);
        }
    }
}
```


```java
public class SnakeTileTest {
    private Player player;
    private SnakeTile tile;

    @Before
    public void before(){
        player = new Player("Steve");
        tile = new SnakeTile(40);
    }

    @Test
    public void snakeTileHasANumber(){
        assertEquals(40, tile.getNumber());
    }

    @Test
    public void snakeTileHasPlayers(){
        assertEquals(player, tile.getPlayers());
    }

    @Test
    public void tileCanUpdatePlayerPosition(){
        tile.addPlayers(player);
        tile.updatePlayerPosition();
        assertEquals(35, player.getCurrentPosition());
    }
}
```

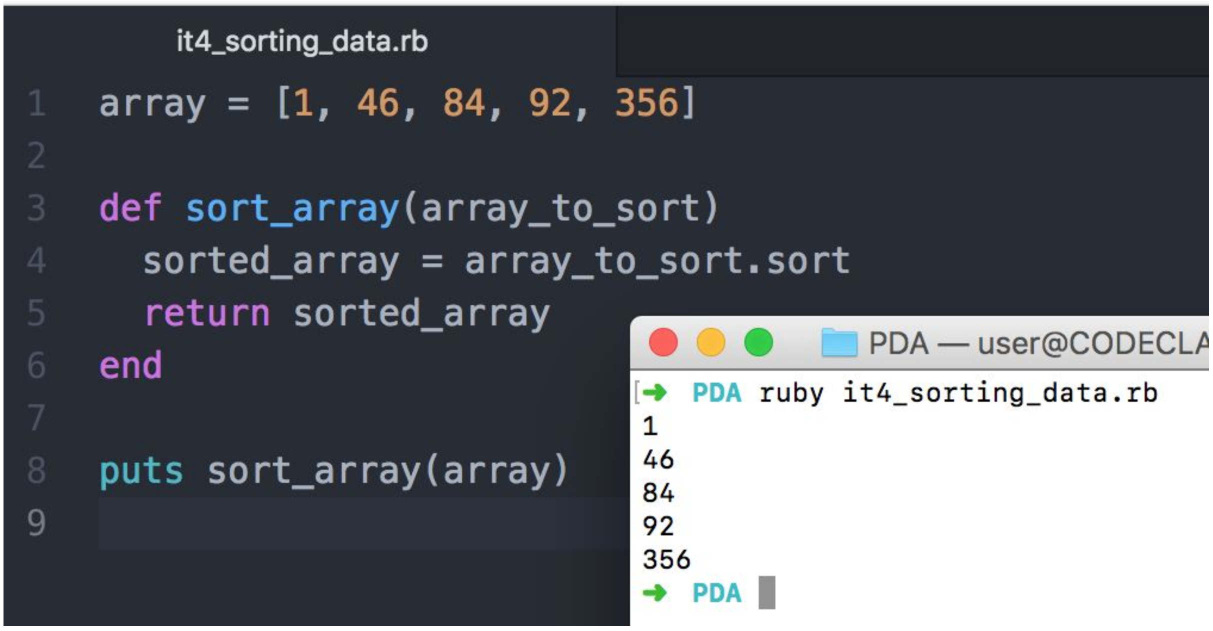
I.T 3 - Example of searching and sorting data.



```ruby
cupboard = ["chilli", "thyme", "oregano", "basil", "cumin"]


def sort_alphabetically(array_name)
  sorted_array = array_name.sort
  return sorted_array
end

def check_if_cupboard_includes(array, needed_ingredient)
  if array.include?(needed_ingredient)
    puts "You have " + needed_ingredient + "! No need to go to the shop!"
  else
    puts "You don't have " + needed_ingredient + "! You need to buy it!"
  end
end

puts sort_alphabetically(cupboard)

puts check_if_cupboard_includes(cupboard, "cumin")
```

```
➜ PDA ruby it3_searching_data.rb
basil
chilli
cumin
oregano
thyme
You have cumin! No need to go to the shop!

➜ PDA
```

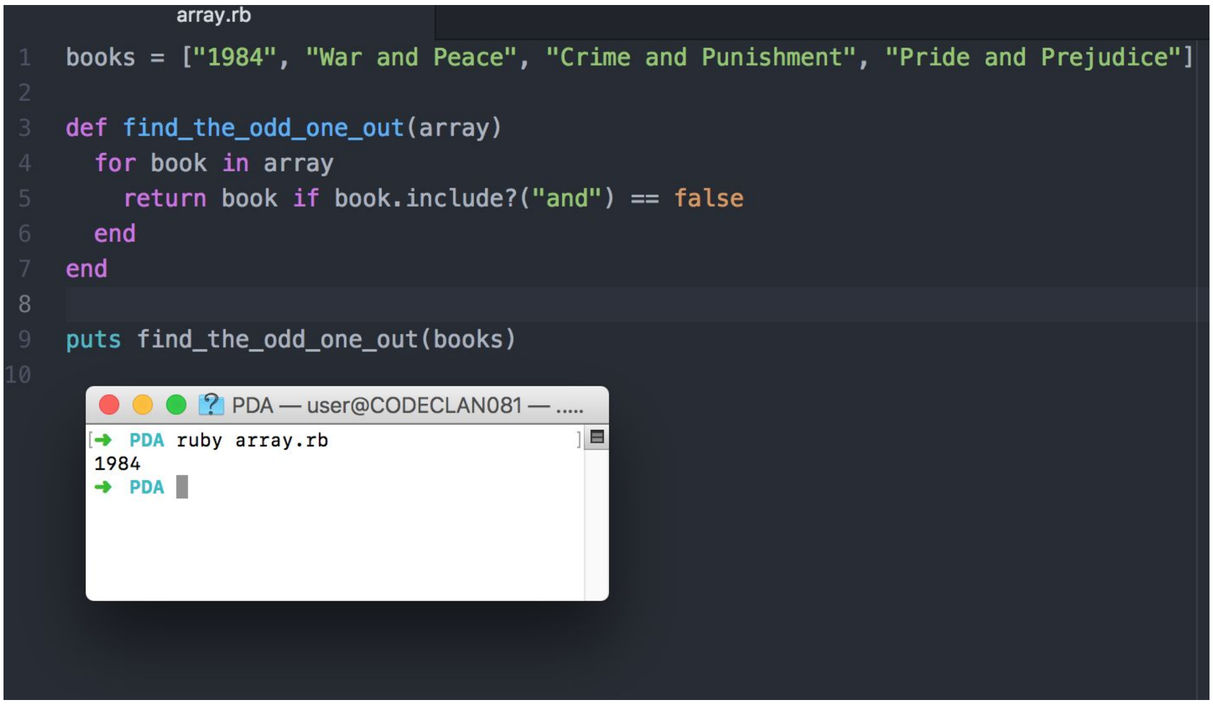
I.T 4 - Example of an array, a function that uses an array and the result



```ruby
array = [1, 46, 84, 92, 356]

def sort_array(array_to_sort)
  sorted_array = array_to_sort.sort
  return sorted_array
end

puts sort_array(array)
```

```
➜ PDA ruby it4_sorting_data.rb
1
46
84
92
356
➜ PDA
```

I.T 5 - Example of a hash, a function that uses a hash and the result



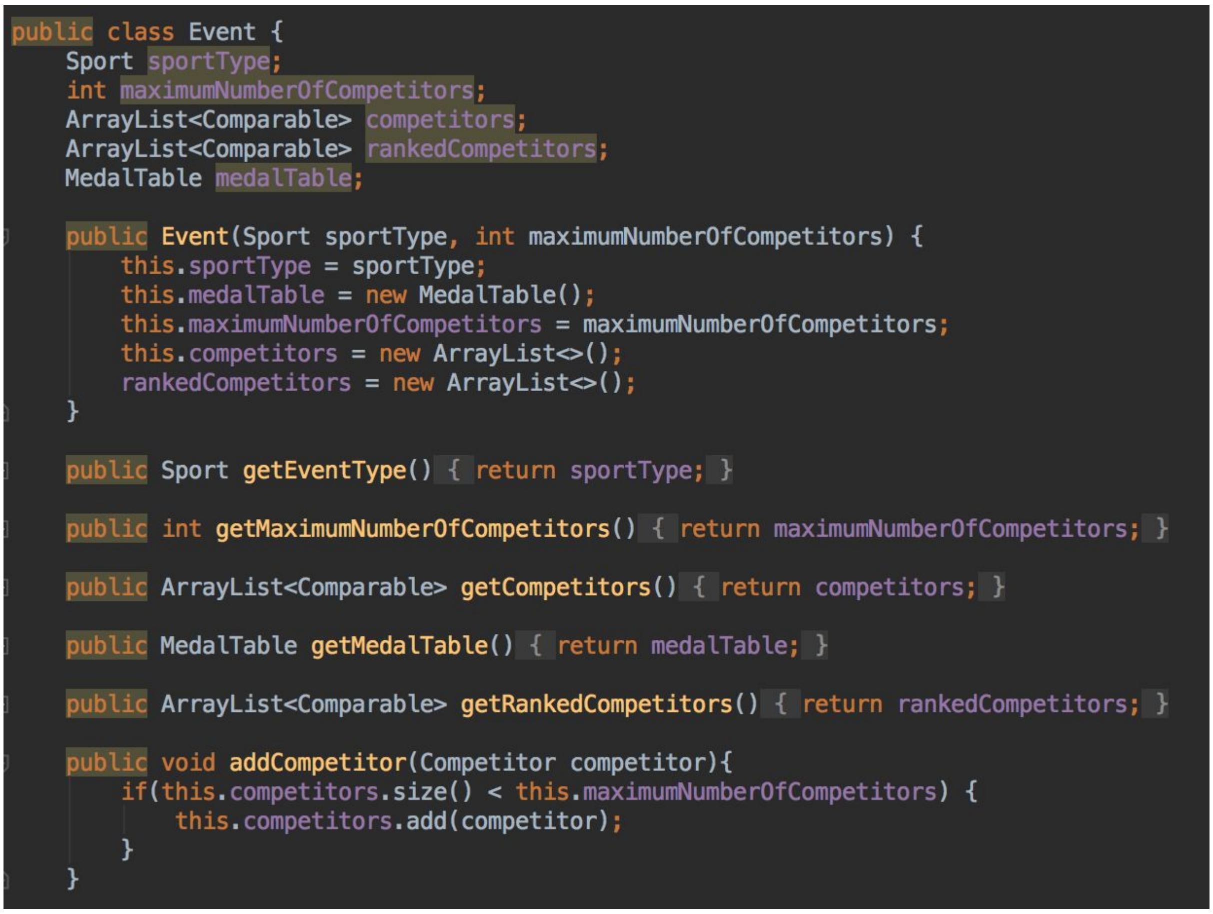
array.rb

```ruby
books = ["1984", "War and Peace", "Crime and Punishment", "Pride and Prejudice"]

def find_the_odd_one_out(array)
  for book in array
    return book if book.include?("and") == false
  end
end

puts find_the_odd_one_out(books)

```

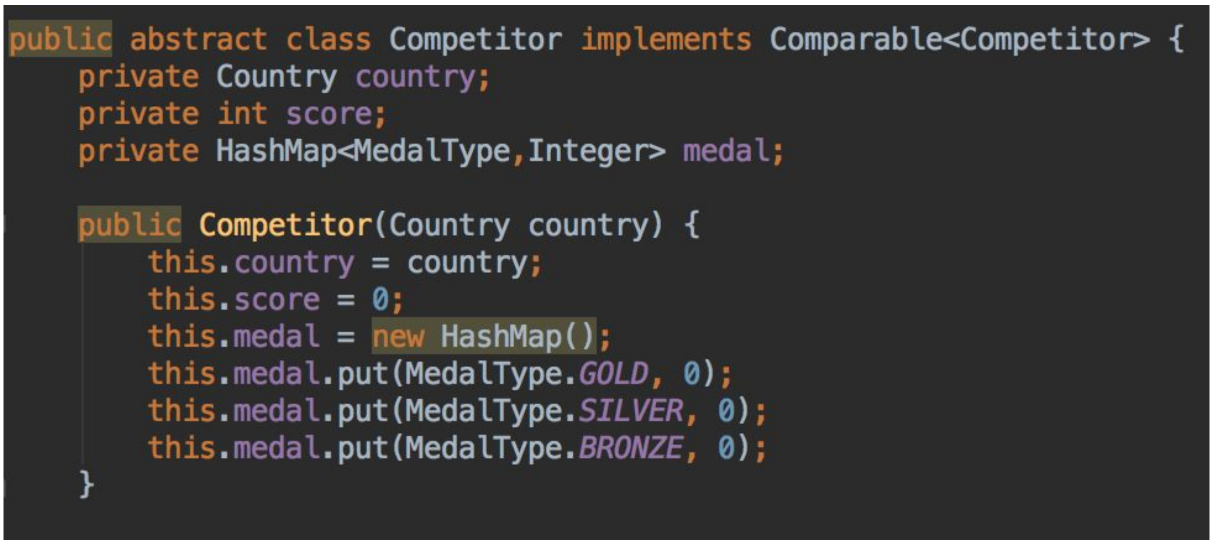
PDA — user@CODECLAN081 — .....

```
➜  PDA ruby array.rb
1984
➜  PDA
```

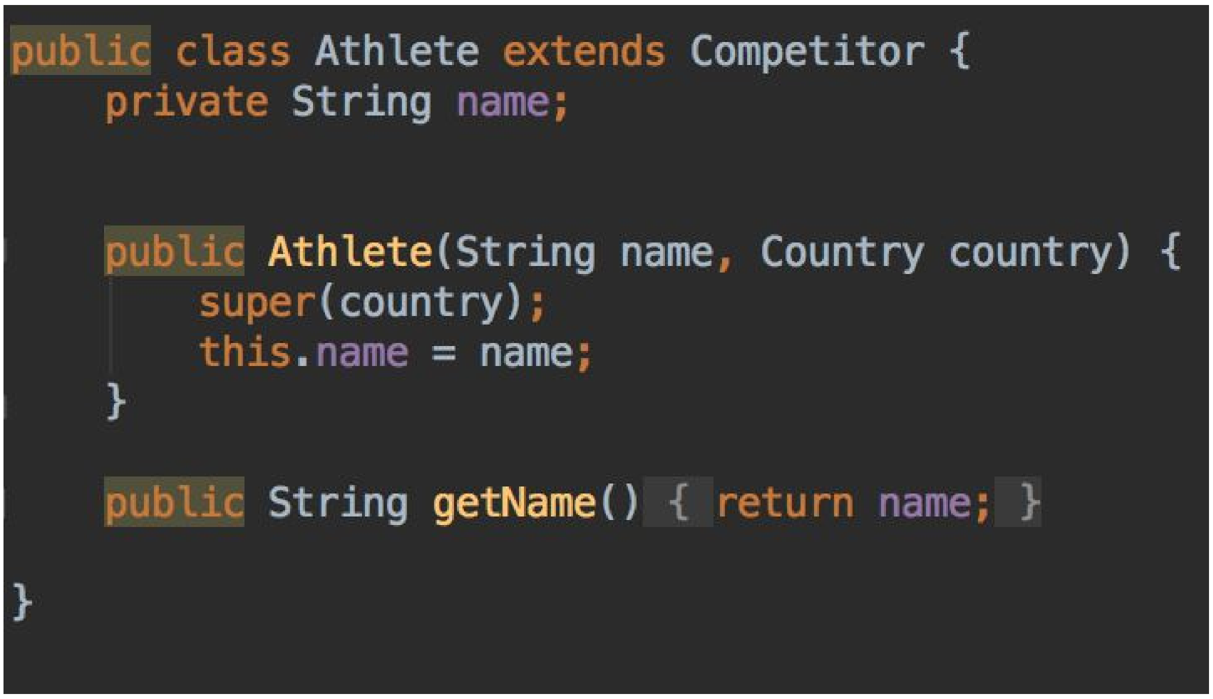
I.T 7 - Demonstrate the use of polymorphism in a program



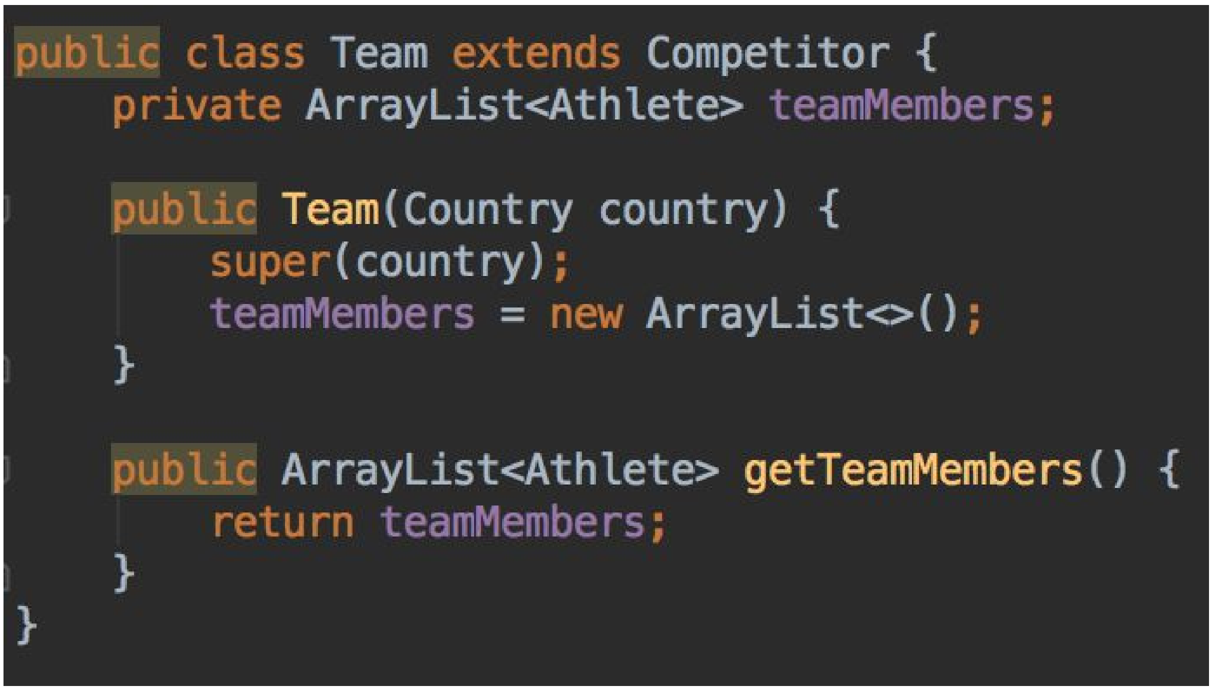
```java
public class Event {
    Sport sportType;
    int maximumNumberOfCompetitors;
    ArrayList<Comparable> competitors;
    ArrayList<Comparable> rankedCompetitors;
    MedalTable medalTable;

    public Event(Sport sportType, int maximumNumberOfCompetitors) {
        this.sportType = sportType;
        this.medalTable = new MedalTable();
        this.maximumNumberOfCompetitors = maximumNumberOfCompetitors;
        this.competitors = new ArrayList<>();
        rankedCompetitors = new ArrayList<>();
    }

    public Sport getEventType() { return sportType; }

    public int getMaximumNumberOfCompetitors() { return maximumNumberOfCompetitors; }

    public ArrayList<Comparable> getCompetitors() { return competitors; }

    public MedalTable getMedalTable() { return medalTable; }

    public ArrayList<Comparable> getRankedCompetitors() { return rankedCompetitors; }

    public void addCompetitor(Competitor competitor){
        if(this.competitors.size() < this.maximumNumberOfCompetitors) {
            this.competitors.add(competitor);
        }
    }
```



```java
public abstract class Competitor implements Comparable<Competitor> {
    private Country country;
    private int score;
    private HashMap<MedalType,Integer> medal;

    public Competitor(Country country) {
        this.country = country;
        this.score = 0;
        this.medal = new HashMap();
        this.medal.put(MedalType.GOLD, 0);
        this.medal.put(MedalType.SILVER, 0);
        this.medal.put(MedalType.BRONZE, 0);
    }
```



```java
public class Athlete extends Competitor {
    private String name;


    public Athlete(String name, Country country) {
        super(country);
        this.name = name;
    }

    public String getName() { return name; }

}
```



```java
public class Team extends Competitor {
    private ArrayList<Athlete> teamMembers;

    public Team(Country country) {
        super(country);
        teamMembers = new ArrayList<>();
    }

    public ArrayList<Athlete> getTeamMembers() {
        return teamMembers;
    }
}
```