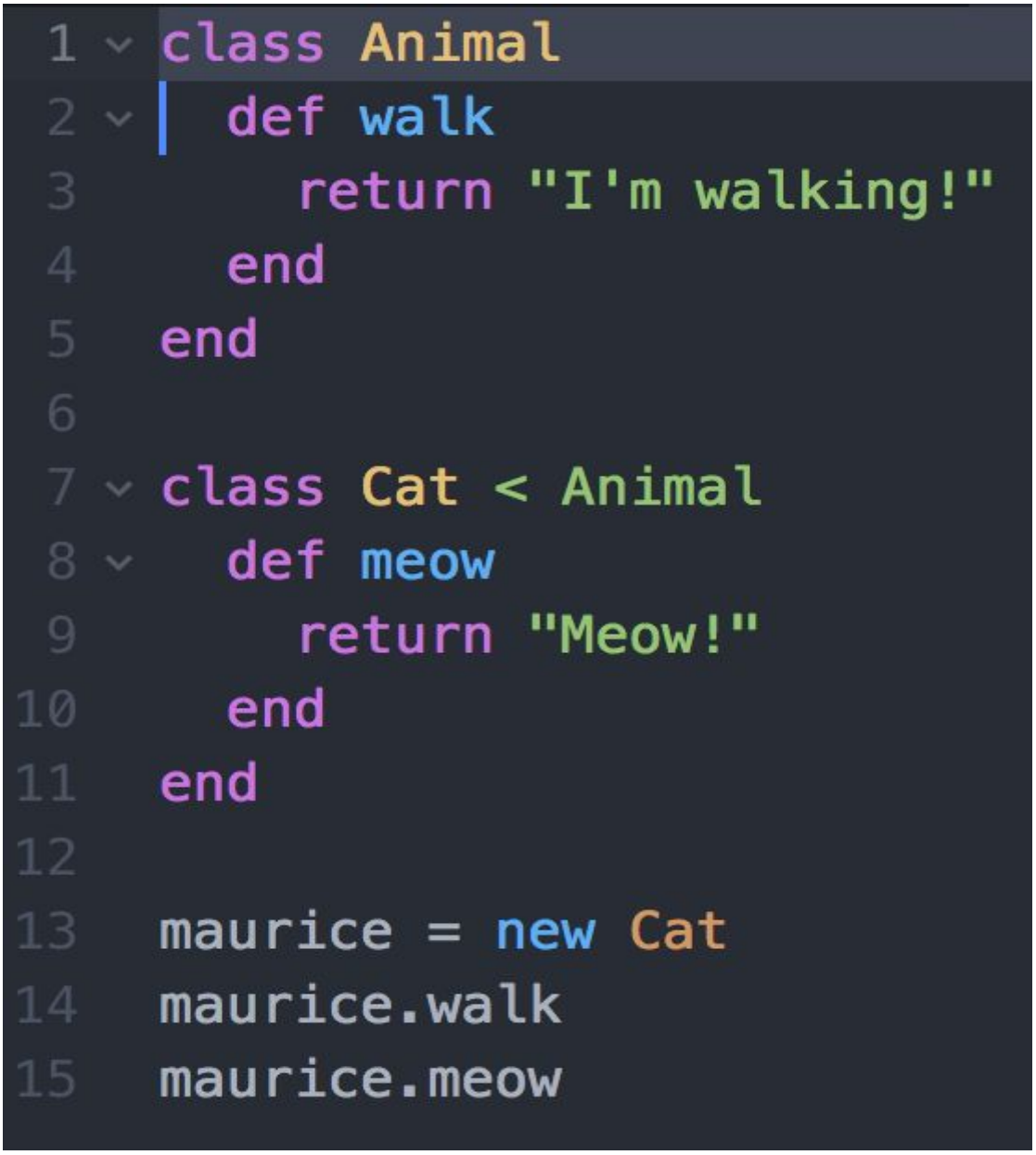
Evidence for Implementation and Testing Unit.

Gabriela Lewandowska
E15
18 September 2017

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

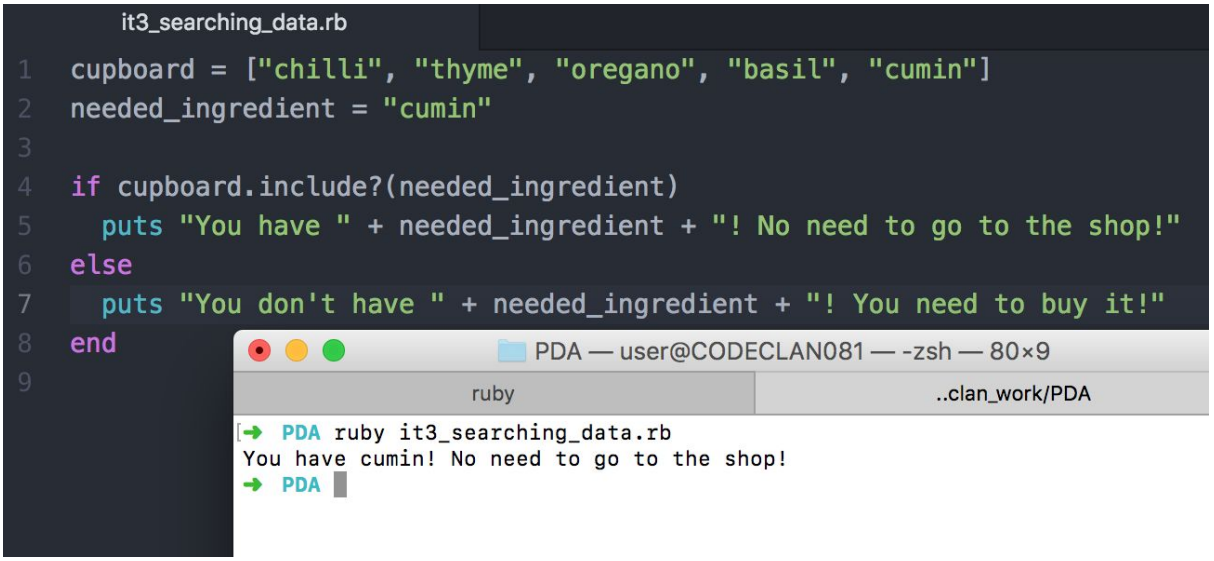
```
public class Tile {
     int number;
     ArrayList<Player> players;

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

    public int getNumber() { return number; }

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

I.T 2 - Inheritance in a program.

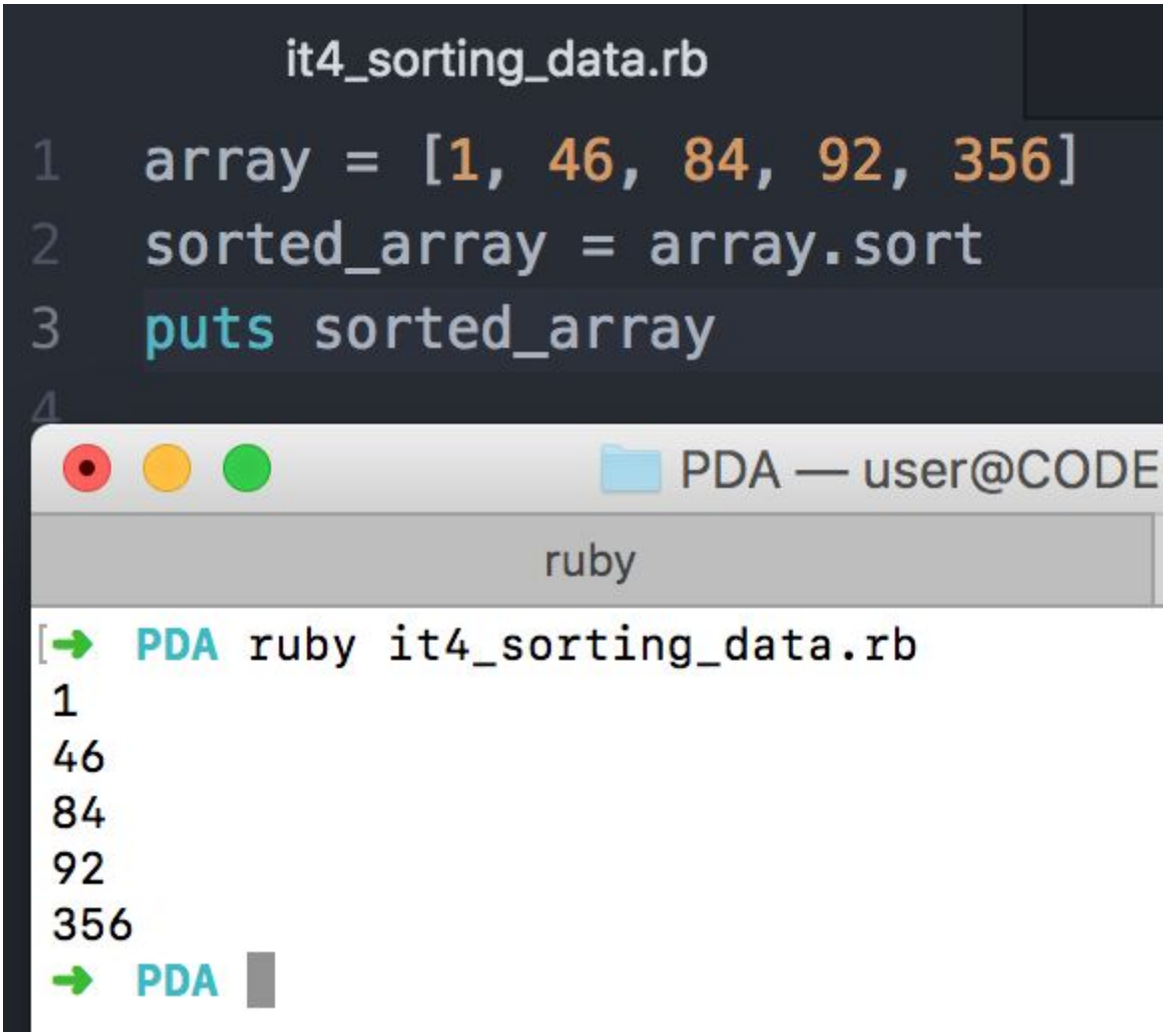


```ruby
class Animal
  def walk
    return "I'm walking!"
  end
end

class Cat < Animal
  def meow
    return "Meow!"
  end
end

maurice = new Cat
maurice.walk
maurice.meow
```

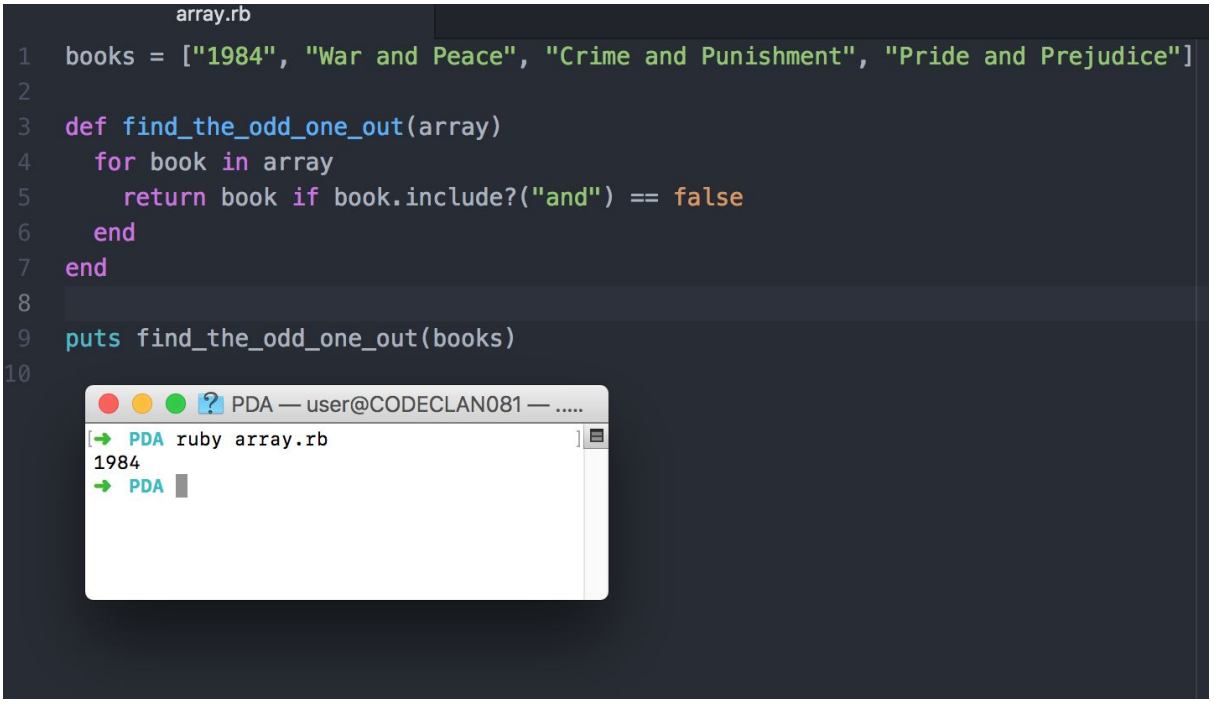
I.T 3 - Example of searching and sorting data.



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



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



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

```java
public class Event {
    Sport sportType;
    int maximumNumberOfCompetitors;
    ArrayList<Competitor> competitors;
    ArrayList<Competitor> 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<Competitor> getCompetitors() { return competitors; }

    public MedalTable getMedalTable() { return medalTable; }

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

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


    public void assignScoreToCompetitors(){
        Random random = new Random();
        for(int i = 0; i < this.competitors.size(); i++){
            this.competitors.get(i).setScore(random.nextInt(100));
        }
    }
```

```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;
    }
}
```