PDA Alice Rees - Implementation & Testing

I.T 1 Encapsulation

Properties inside the class are private and have getter/setter methods:

```
public class Blackjack implements Game, Serializable {
    private Dealer dealer;
    private ArrayList<Playable> participants;
    private Player[] players;
    private ArrayList<Card> deck;
    private HashMap<Playable, Integer> cardValues;
    private ArrayList<Playable> outParticipants;
    private ArrayList<Playable> blackjackParticipants;

    public Dealer getDealer() {
        return dealer;
    }

    public void setDealer(Dealer dealer) {
        this.dealer = dealer;
    }
```

I.T 2 Use of inheritance

A class

```
Pet
public abstract class Pet {
    int numLegs = 4;
    String name;

    public Pet(int numLegs, String name) {
        this.numLegs = numLegs;
        this.name = name;
    }

    public int getNumLegs() {
        return numLegs;
    }

    public String getName() {
        return name;
    }

    public String goToSleep() {
        return "zzzzz";
    }

    public abstract String makeNoise();
}
```

A class which inherits from this class

```java
public class Cat extends Pet {

    public Cat(String name){
        super(4, "Kipper");
    }

    public String goToSleep() {
        return this.name + " curls up and goes " + super.goToSleep();
    }

    @Override
    public String makeNoise() {
        return "purr";
    }

}
```

An object in this class

```java
public void before() {
    cat = new Cat("Kipper");
}
```

A method using info inherited from another class

```java
    @Test
    public void testSleeping() {
        assertEquals("Kipper curls up and goes zzzzz", cat.goToSleep());
    }
```

I.T 3 Demonstrate searching data in a program.

Function which searches data (input name, output age):

```ruby
@people = [
  {name: "Alice",
  age: 27,
  location: "Edinburgh"},
  {name: "Sam",
  age: 28,
  location: "Leeds"},
  {name: "Dan",
  age: 25,
  location: "London"}
]


def find_age_by_name(name)
  person_hash = @people.select {|person| person[:name] == name}
  return person_hash[0][:age]
end

puts find_age_by_name("Alice")
```

Result of that function (input name Alice, output age 27):

```
➜  evidence git:(master) ✗ ruby searching_sorting.rb
27
```

I.T 4 Demonstrate sorting data in a program

Function which sorts data:

```ruby
@people = [
  {name: "Alice",
  age: 27,
  location: "Edinburgh"},
  {name: "Sam",
  age: 28,
  location: "Leeds"},
  {name: "Dan",
  age: 25,
  location: "London"}
]


def sort_by_age(array)
  sorted = array.sort_by {|k| k[:age]}
  return sorted
end

puts sort_by_age(@people)
```

Result of that function (people sorted by age)

```
➜  evidence git:(master) ✗ ruby searching_sorting.rb
{:name=>"Dan", :age=>25, :location=>"London"}
{:name=>"Alice", :age=>27, :location=>"Edinburgh"}
{:name=>"Sam", :age=>28, :location=>"Leeds"}
```

I.T 5 Demonstrate use of an array in a program.

An array and an array being used in a function:

arrays_and_hashes.rb

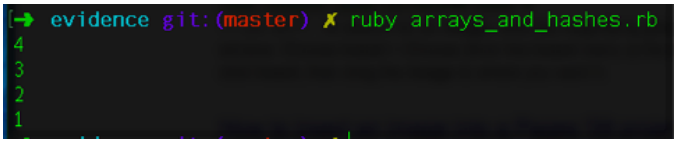
```ruby
my_array = [1, 2, 3, 4]

# Reverse the order of the array

def reverse(array)
  return array.reverse
end

puts reverse(my_array)
```

Function Outcome



I.T 6 Demonstrate use of a hash in a program.

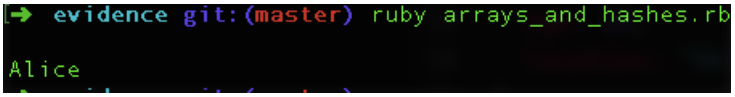
A hash and a hash being used in a function:

```ruby
my_hash = {
  name: "Alice",
  age: 26,
  location: "Edinburgh"
}

def get_name(hash)
  return hash[:name]
end

puts get_name(my_hash)
```

Function outcome:



I.T 7 Demonstrate the use of Polymorphism in a program

This ‘getWinner’ function uses the Interface ‘Playable’ to allow Dealer and Player to win:

```java
public ArrayList<Playable> getWinner() {
    int dealerScore = this.dealer.checkTotal();
    ArrayList<Playable> winners = new ArrayList<>();
    if (this.dealer.getBlackjack() == true) {
        winners.add(dealer);
        return winners;
    }
    if (outParticipants.contains(dealer)){
        winners.addAll(blackjackParticipants);
        winners.addAll(participants);
        return winners;
    }
    for (Playable player : participants) {
        int playerScore = player.checkTotal();
        if (playerScore > dealerScore) {
            winners.add(player);
        }
        if (winners.size() == 0){
            winners.add(dealer);
        }
        return winners;
    }
    return winners;
}
```

The Playable interface:

```
public interface Playable {
    void addCards(Card...cards);
    int checkTotal();
    ArrayList<Card> getCards();
    void setBlackjack(Boolean blackjack);
    Boolean getBlackjack();
}
```

Player Class:

```
public class Player implements Serializable, Playable {
    private String name;
    private ArrayList<Card> playerCards;
    private Boolean blackjack;
```

Dealer Class:

```
public class Dealer implements Playable {
    private ArrayList<Card> dealerCards;
    private Boolean blackjack;
```