Exercise 1: My Favorite People (Arrays of Objects)

Assume that the following Person class has been defined.

```java
public class Person {
    private String name;
    private int age;
    private String eyeColor;

    // constructor
    public Person(String aName, int anAge, String aColor) {
        name = aName;
        age = anAge;
        eyeColor = aColor;
    }

    // accessor methods
    public String getName() { return name; }
    public int getAge() { return age; }
    public String getEyeColor() { return eyeColor; }
}
```

First, choose four people in your group and draw an array called `people` that contains four Person objects representing those four people (use strings with all capital letters for the names and eye colors).

Now (in groups of two or three) play a game of memory using one of the sets of cards. One pack in each set has a Java expression. The other pack in each set has a value. Start with all of the cards upside down. When it’s your turn, you turn over one card of each color. If they match you keep those two cards, and keep going; otherwise, your turn is over.

The game ends when you run out of expression cards or the remaining value cards don’t match any expression cards.

Whoever has the most cards wins!
Exercise 2: You’re Driving me Crazy! (Simulating Java Code)

This exercise will help you to understand what happens when objects are declared and created, and when methods are called.

It will also help you to understand the difference between copying from one variable to another when the variable is an object, and when it is a primitive type (int, double, boolean, etc), as well as the difference between changing the values of variables that are objects vs primitive types.

First, take a look at the Car class defined on the next page.

Now execute the code fragment below; let one person play the role of each variable (myCar, yourCar, anotherCar, oldSpeed, and currSpeed), and also have one person be the Java memory manager. When a variable is assigned to, or one of its methods is called, the person playing the role of that variable should act out the effects of the assignment or call. The memory manager should handle calls to new and should also reclaim any memory that gets freed.

```java
Car myCar, yourCar, anotherCar;
int oldSpeed, currSpeed;

myCar = new Car("beep");
yourCar = new Car("honk");
anotherCar = myCar;

currSpeed = myCar.getCurrSpeed();
yourCar.increaseSpeed(7);
anotherCar.increaseSpeed(20);
currSpeed = myCar.getCurrSpeed();

myCar.blowHorn(2);
yourCar.blowHorn(3);
anotherCar.blowHorn(4);

oldSpeed = currSpeed;
myCar = yourCar;
currSpeed = myCar.getCurrSpeed();

myCar.changeSound("ooga");
myCar.blowHorn(currSpeed/5);
yourCar.blowHorn( yourCar.getCurrSpeed()/2 );
anotherCar.blowHorn( myCar.getCurrSpeed()/10 );
anotherCar = myCar;
```
public class Car {
    private int currSpeed;
    private String hornSound;

    public Car(String sound) {
        currSpeed = 0;
        hornSound = sound;
    }

    public void changeSound(String newSound) {
        hornSound = newSound;
    }

    public void blowHorn(int numTimes) {
        while (numTimes > 0) {
            System.out.println(hornSound);
            numTimes--;
        }
    }

    public void increaseSpeed(int milesPerHour) {
        currSpeed = currSpeed + milesPerHour;
    }

    public int getCurrSpeed() {
        return currSpeed;
    }
}
Exercise 3: Match em Up (Types)

Every Java expression has a type, and the better you understand types the easier you’ll find it to write correct code and to understand other people’s code.

Test your understanding of types by matching up the cards from two sets: one has code fragments from the previous Car exercise, and the other has Java types.

Exercise 4: ArrayLists of People (Writing code)

For this exercise, we’ll reuse the People class from Exercise 1, but this time we’ll store people in an ArrayList instead of an array.

Your job is to write a complete Java program that works as follows:
1. Ask the user how many people there are.
2. Read information about that many people (name, age, eye color), storing the information in an ArrayList. The ArrayList should be sorted by age (so the youngest person is in position 0, the next youngest is in position 1, and so on).
3. Write out the information about each person (from youngest to oldest).

Start by writing an outline of the main method (in English). What should it do first? Then what?

Then decide how to add a new person to the ArrayList of people in the correct place (so that the list is sorted by age).

Finally, decide in what order to write the code, so that you can test it as you go along.
Exercise 5: Hats and Snacks (Logical Thinking)

Puzzle 1: Four people are standing in a row, one behind the other, all facing the same way with person 1 at the front and person 4 at the end. Between numbers 1 and 2 is a wall that prevents any of the others from seeing person 1. So number 4 can see numbers 3 and 2, number 3 can see number 2, and numbers 2 and 1 cannot see anyone.

Problem: The people are told that two of them are wearing white hats and two are wearing black hats. They are told that if any one of them can say what color hat they are wearing, they will all win fabulous prizes! They are not allowed to communicate with each other in any way; all they can do is say “No, I don’t know what color my hat is”, or “Yes I know!” Will anyone be able to win the fabulous prizes? If so, who and how?

Puzzle 2: During the lunch hour at school, a group of five boys from Miss Smith’s homeroom visited a nearby snackbar. One of the five boys took a candy bar without paying for it. When the boys were questioned by the school principal, they made the following statements:

Rex: "Neither Earl nor I did it."
Jack: "It was Rex or Abe."
Abe: "Both Rex and Jack are lying."
Dan: "Abe’s statement is not true."
Earl: "What Dan said is wrong."

When Miss Smith was consulted, she said, "Three of these boys always tell the truth, and the other two always lie."

Problem: Assuming that Miss Smith is correct, determine who took the candy bar.
people[1].getAge()

(people[0].getEyeColor().compareTo(people[3].getEyeColor())) < 0

people[people.length-1].getAge() > 18

people[2].getName().length() < 6;

(people[2].getName().substring(0,1)).equals((people[3].getEyeColor().substring(0,1)))

people[1].getEyeColor().charAt(1)
### Value Cards (exercise 1)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

| L | R | A |
yourCar

"honk"

yourCar.getCurrSpeed()

new Car("beep")

yourCar.getCurrSpeed()/2

yourCar.blowHorn(3)

numTimes > 0

currSpeed + milesPerHour

currSpeed = 0
Type Cards (Exercise 3)

Car

String

int

Car

int

void

boolean

int

int