Problem Solving Session

- The remainder of today's class will comprise the ***problem solving session*** (***PSS***).
- Your instructor will divide you into ***teams of 3 or 4 students***.
- Each team will ***work together*** to solve the following problems over the course of ***20-30 minutes***.
  - You may work on paper, a white board, or digitally as determined by your instructor.
  - You will submit your solution by pushing it to GitHub before the end of class.
- Your instructor will go over the solution before the end of class.
- If there is any time remaining, you will begin work on your homework assignment.

Class participation is a significant part of your grade (20%). This includes in class activities and the problem solving session.

Your Course Assistants will grade your participation by verifying that you pushed your solutions before the end of the class period each day.

Problem Solving Team Members

Record the name of each of your problem solving team members here.

Do not forget to ***add every team member's name***! Your instructor (or course assistant) may or may not use this to determine whether or not you participated in the problem solving session.

Dessa Shapiro
Mason Hendricks
Chris Holmes

Moonbase Alpha

Moonbase Alpha is composed of a number of hubs. Two of those hubs, named Hyperion and Odyssey, are connected by a tunnel. Transport vehicles called Rovers are used to transport personnel and cargo between the two hubs. There is only a single tunnel that connects these two hubs, but fortunately it is large enough to allow several Rovers to traverse the tunnels in both directions simultaneously.

Problem Solving 1

Any number of Rovers can traverse the tunnel at the speed that their load allows at the same time.

In the class that will be created to represent a Rover, we'll need a good way to handle the hub names. For now, write only the inner class that should be used for this purpose.

```
public String leavingName(Hub hub) {
      Return hub.getName();
}
public String arrivingName(Hub hub) {
      Return hub.getName();
}
Public class Rover {
Public enum HubName{
HYPERION,
ODYSSEY;
}
Private HubName name; //inner class start
Private Hub(HubName name){
this.name = name;
}; //inner class end?

Private Hub location;
Private Hub(String name){
this.

Public Rover(Hub location){
This.location = location;
}
}
@Override
Public String toString() {
      return hubName;
}
```

```
public class Rover implements Runnable{
      //inner class here

      public static final int SECOND = 1000;
      private String name;
      private int transitTime;
      private Hub startHub;
      private Hub endHub;

      public Rover(String name, int transitTime, Hub
startHub, Hub endHub) {
      this.name = name;
      this.transitTime = transitTime;
      this.startHub = startHub;
      this.endHub = endHub;
}

}
```

Problem Solving 2

Each Rover has a:

- Name
- Transit time (in seconds)
- Starting hub
- Ending hub

All of the Rovers can traverse the tunnel at the speed that their load allows at the same time.

Create a new class to represent a Rover. It should include all of the specified state and be able to run independently from other Rovers.

Show the inner class from Problem 1 as a comment. Do not worry about the tunnel crossing logic - stub out any methods for it.

Problem Solving 3

Let's look at a single Rover named Orion who takes 3 seconds to transit the tunnel to Odyssey. If Orion was the **only** Rover that wanted to transit the tunnel, the output would look like:

```
Orion(3) is ready to start traversal at the
Hyperion end of the tunnel.
Orion(3) is starting traversal.
     Orion(3): 1 second.
     Orion(3): 2 seconds.
Orion(3) has completed traversal and has
reached the Odyssey end of the tunnel.
```

On the left write what you think the output might look like if the following 3 Rovers all wanted to cross the tunnel.

1. Orion - 3 seconds to Odyssey
2. Nebula - 5 seconds to Hyperion
3. Quantum - 4 seconds to Odyssey

```
Quantum(4) is ready to start traversal at the
Hyperion end of the tunnel.
Orion(3) is ready to start traversal at the
Hyperion end of the tunnel.
Nebula(5) is ready to start traversal at the
Odyssey end of the tunnel.
Quantum(4): 1 second.
Orion(3): 1 second.
Nebula(5): 1 second.
Quantum(4): 2 seconds.
Orion(3): 2 seconds.
Nebula(5): 2 seconds.
Quantum(4): 3 seconds.
Orion(3) has completed traversal and has reached
the Odyssey end of the tunnel.
Nebula(5): 3 seconds.
Quantum(4) has completed traversal and has reached
the Odyssey end of the tunnel.
Nebula(5): 4 seconds.
Nebula(5) has completed traversal and has reached
the Odyssey end of the tunnel.
```

Problem Solving 4

```
@Override
    public void run() {
        int i = 0;
        sout(this.name+“is ready to start traversal at
the” + this.starting + “end of the tunnel.”)
            while(i<this.trasitTime) {
                if(i == 0) {
                    System.out.println(this.name+"is
                    starting traversal");}
                 else {
                    System.out.println(this.name + i + “
                    seconds.”);
                  }
                  try {
                      Thread.sleep(SECOND);
                       i++;
                  }catch(InterruptedException e) {}
            }
            System.out. println(this.name + “has
completed traversal and has reached the ” + hub.getName()
+ “ end of the tunnel”)

}
```

Write the run method for a Rover. It should print:

- The message when it arrives at the tunnel.
- The starting message at time 0.
- The crossing message after each second they have been in the tunnel.
- The exit message when their transit time is complete.

Example Format:

```
Orion(3) is ready to start traversal at the
Hyperion end of the tunnel.
Orion(3) is starting traversal.
     Orion(3): 1 second.
     Orion(3): 2 seconds.
Orion(3) has completed traversal and has
reached the Odyssey end of the tunnel.
```