Oregon Trail

Write a program **OregonTrail.java** that runs a game or "season" of the game, Oregon Trail. First, here are the rules of the game. A wagon starts out on the Oregon Trail, which is 2000 miles long, on March 1. It must reach the end of the trail by the end of the year, and if it does, the player wins the game. There are three ways to lose the game:

- If still out on the trail on January 1, winter cold will overcome the travelers and they will die.
- The wagon is stocked with 500 pounds of food, and the travelers consume 5 pounds of food each day. If they run out of food, they will starve.
- Travelers have a health rating of 5 when they start out. On some days (randomly, one day in 15) they get sick, meaning their health rating drops by one. If their health rating drops to 0, they die of sickness.

The game is played in a series of turns. On each turn, the player chooses to travel, to hunt, or to rest. Traveling takes between 3 and 7 days, randomly chosen, and moves the wagon between 30 and 60 miles up the trail. Hunting takes between 2 and 5 days, and gives the travelers 100 pounds of fresh food. Resting takes between 2 and 5 days, and improves the travelers' health by 1 point.

Your program should carry out the complete game, from start to finish, and should implement logic by which the player makes decisions about what to do on a given turn. You can fiddle with this, but a decent strategy is, for each turn, if health is below 3, then rest; otherwise, if food is below 100, then hunt; otherwise, travel.

Your completed program should print a play-by-play of the game progress to standard output: for each turn, what is the date, how far do we still have to go, how much food do we have, and what is the current health rating? Also print out what actions are being taken – travel, hunt, rest – and note whenever sickness strikes. Your program should, before terminating, state what happened: either the player made it, or they got sick, or they starved, or they froze to death.

Here are some tips for implementing the program:

- Both as good general practice and for your own sanity, start by declaring a bunch of variables with descriptive names that capture all of the metrics of the game rules. Make these **static** and **final**, because they are constants. For example:

```
public static final int TRAVEL_DAYS_MIN = 3;
public static final int TRAVEL_DAYS_MAX = 7;
public static final int FOOD_PER_DAY = 5;
```

- Other variables will track progress through the game. Declare these as **static**, but not **final**. Most likely you'll want **month**, **day**, **milesToGo**, **food**, and **health**.
- Break the logic of playing the game down into small, reusable pieces, and implement each in a **public**, **static** method. Some methods then will depend on making calls to other methods, and you can build up the full game logic in this way. Here are some suggestions for methods that might serve well:

```
public static int randomNumber(int from, int to)
public static void passOneDay()
public static void travel()
public static void hunt()
public static void rest()
public static void playOneTurn()
public static void showStatus()
```

- You'll need a little special conditional logic to deal with the irregular lengths of months in the calendar. Keep this simple – just test if the month number is one of the few that have 30 days, and remember you don't have to mess with February, or leap years, any of that: it's always 30 or 31.
- You can use ThreadLocalRandom to generate a random integer in a desired range. For example to get an integer between 3 and 7:

```
import java.util.concurrent.ThreadLocalRandom; // after your package statement

int days = ThreadLocalRandom.current().nextInt(3, 8) // yes, 8 – it's "exclusive"
```