Design comments

General Comments:

- Naming is very good (for example intervalMinutes tells me that this is a time interval in minutres units)
- Your ExamRoom should implement Comparable<T extends IExamRoom>, thus you will NOT need to cast in compareTo. The same with Patient. Please also verify that you cannot have null objects (either Patient or ExamRoom) as inputs to your compareTo methods. Otherwise you need to implement null-proof compareTo.
- You should code to interfaces, including those you define. You do this with IPriorityQueue, which is good. So, if you have IPatient and IEmergencyRoom you should also use these as compile-time types instead of Patient and Emergency Room.
- You should check the correctness of input of user. For example, here is how I managed to crash your programm

  Enter 0/1 to choose simulation type(random/preset): a
  Exception in thread "main" java.util.InputMismatchException
  at java.util.Scanner.throwFor(Scanner.java:864)
  at java.util.Scanner.next(Scanner.java:1485)
  at java.util.Scanner.nextInt(Scanner.java:2117)
  at java.util.Scanner.nextInt(Scanner.java:2076)
  at edu.neu.ccs.cs5010.ERSimulator.setUpParameter(ERSimulator.java:271)
  at edu.neu.ccs.cs5010.ERSimulator.main(ERSimulator.java:181)
  at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
  at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
  at
  sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
  at java.lang.reflect.Method.invoke(Method.java:498)
  at com.intellij.rt.execution.application.AppMain.main(AppMain.java:147)

  It would be better to allow user to choose preset/random rather than 0/1
- Tests - should have different test method per different test

Abstraction and Modularity:

- You should have PatientGenerator, to remove this responsibility from ERSimulator
- Preset and Random modes – currently you work **procedurally** (if it is Present run one method, generatePresetPatient, if it is Random run second method, generateRandomPatient). You should try to think in Object Oriented way. That is, you define IPatientGenerator which is one of: RandomGenerator and PresetGenerator. Once you identify an itemization of this kind it means that you should have two concrete implementations. Your PatientGenerator can be the abstract class that operates common operations for both and everything else should be hidden behind

inheritance and overriding. Having this will eliminate code duplication, that you currently have in your generatePresetPatient and generateRandomPatient.

- You should consider moving your printing from ERSimulator to another module (Printer ?)
- Maybe moving Analysis of ERSimulator to Analysis?
- Urgency should be abstracted into a different class.

Correctness Comments from TAs:

1. Tests in Stack and Queue fail when run individually.
2. Priority Queue tested well.