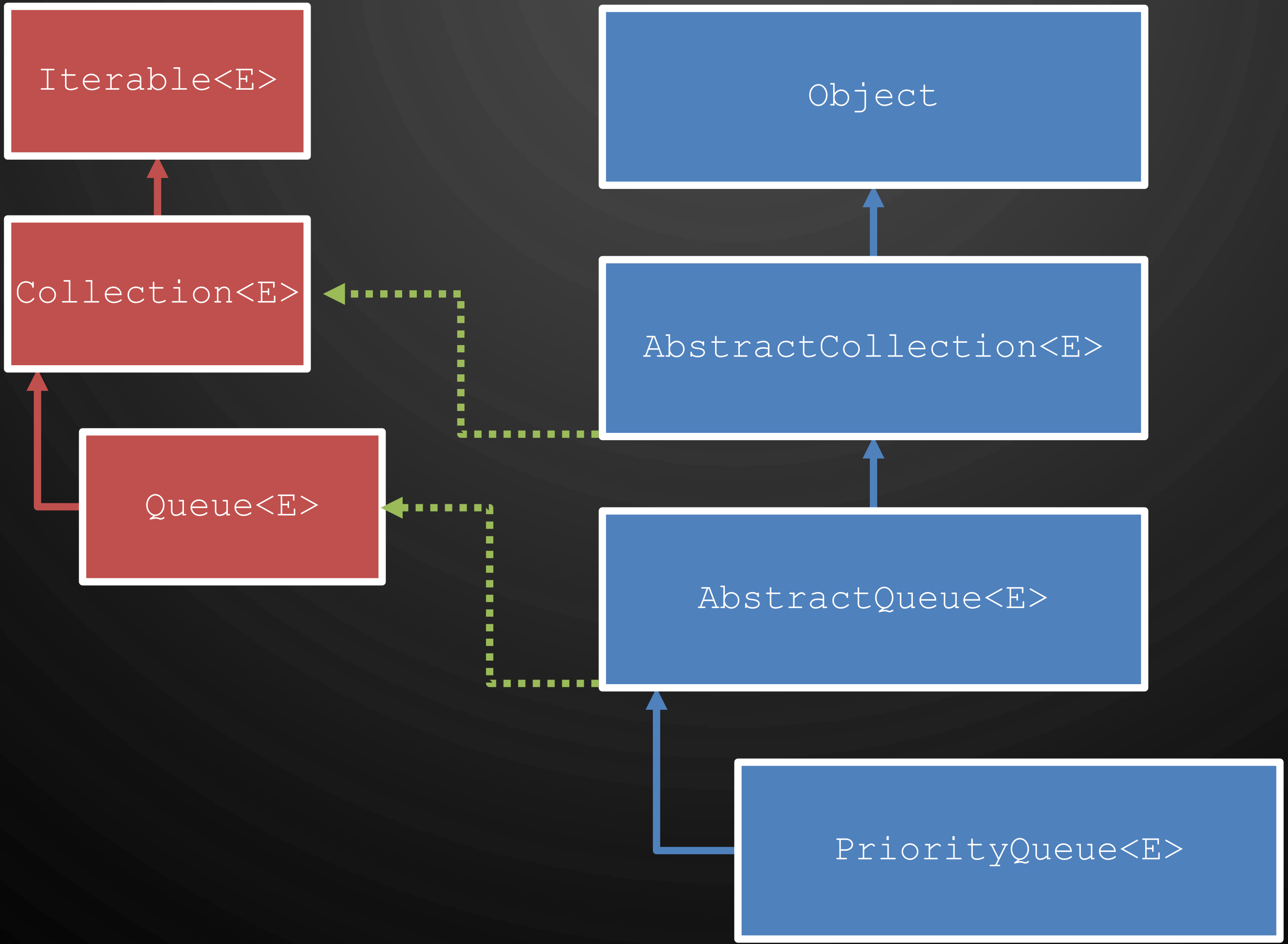
JAVA PRIORITY QUEUE

SUMMARY OF CLASSES (PRIORITY QUEUE RELATED)

- `PriorityQueue<E>` - array-based heap implementation of minimum priority queue
- `Comparator<E>` - can be useful for defining your own comparison between objects

- Others outside the scope of this course
- To find how to use them, go to the Java API!

Interfaces
Classes



EXAMPLE OF USING PRIORITYQUEUE<E>

```java
Scanner s = new Scanner(new File(“numbers.txt”));
PriorityQueue<Integer> numbers = new PriorityQueue<>();
while(s.hasNextInt())
  numbers.add(s.nextInt());
…elsewhere…
int sum = 0;
while(!numbers.isEmpty())
  sum += numbers.poll(); //poll is removeMin()
```

DEFINING A COMPARATOR

- First method - No new class and simply override **Object**.compareTo(**Object** o) in any class
- Second – separate comparator class that implements **Comparator<E>** interface
  - Must define compare(**E** o1, **E** o2) and equals(**Object** o)
    - Here equals is a comparison to another comparator

PROBLEM

- Event driven simulation – you want to estimate the profit for a coffee shop. There is an input file online stating the number of seats in the shop, the price per cup of coffee, and arrive events with a given time (integer) and number of partisans (integer) (1 pair per line)
- Use a priority queue of events, ordered by time to see how much profit the store will earn over this period. Rules:
  - Arrive event - If a group enters and there are not enough seats they will leave. If they stay, an order event will be created at the current time + 1 + a random number below 4
  - Order events - Every partisan of the group will buy 1 or 2 cups of coffee. Each orderEvent will also spawn a leaveEvent at the currentTime + 1 + a random number below 10.
  - Leave event – When a group leaves, their chairs are opened up to another group
- Create an object oriented solution to this problem with your team. PLAN-IMPLEMENT-TEST!