```python
#Classes Challenge 39:  Epidemic Outbreak Terminal App
import random

class Simulation():
    """A class to control the simulation and help facilitate in the spread of the disease."""

    def __init__(self):
        """Initialize attributes"""
        self.day_number = 1

        #Get simulation initial conditions from the user
        print("To simulate an epidemic outbreak, we must know the population size.")
        self.population_size = int(input("---Enter the population size: "))

        print("\nWe must first start by infecting a portion of the population.")
        self.infection_percent = float(input("---Enter the percentage (0-100) of the population to initially infect: "))
        self.infection_percent /= 100

        print("\nWe must know the risk a person has to contract the disease when exposed.")
        self.infection_probability = float(input("---Enter the probability (0-100) that a person gets infected when exposed to the disease: "))

        print("\nWe must know how long the infection will last when exposed.")
        self.infection_duration = int(input("---Enter the duration (in days) of the infection: "))

        print("\nWe must know the mortality rate of those infected.")
        self.mortality_rate = float(input("---Enter the mortality rate (0-100) of the infection: "))

        print("\nWe must know how long to run the simulation.")
        self.sim_days = int(input("---Enter the number of days to simulate: "))


class Person():
    """A class to model an individual person in a population."""

    def __init__(self):
        """Initialize attributes"""
        self.is_infected = False #Person starts healthy, not infected
        self.is_dead = False #Person starts ALIVE
        self.days_infected = 0 #Keeps track of days infected for individual person


    def infect(self, simulation):
        """Infect a person based on sim conditions"""
        #random number generated must be less than infection_probability to infect
        if random.randint(0, 100) < simulation.infection_probability:
            self.is_infected = True


    def heal(self):
        """Heals a person from an infection"""
        self.is_infected = False
        self.days_infected = 0


    def die(self):
        """Kill a person"""
        self.is_dead = True


    def update(self, simulation):
        """Update an individual person if the person is not dead.  Check if they are infected
           If they are, increase the days infected count, then check if they should die or be healed."""
        #Check if the person is not dead before updating
        if not self.is_dead:
            #Check if the person is infected
            if self.is_infected:
                self.days_infected += 1
                #Check to see if the person will die
                if random.randint(0, 100) < simulation.mortality_rate:
                    self.die()
                #Check if the infection is over, if it is, heal the person
                elif self.days_infected == simulation.infection_duration:
                    self.heal()


class Population():
    """A class to model a whole population of Person objects"""

    def __init__(self, simulation):
        """Initialize attributes"""
        self.population = [] #A list to hold all Person instances once created

        #Create the correct number of Person instances based on the sim conditions
        for i in range(simulation.population_size):
            person = Person()
            self.population.append(person)


    def initial_infection(self, simulation):
        """Infect an initial portion of the population."""
        #The number of people to infect is found by taking the pop size * infection percentage
        #We must round to 0 decimals and cast to an int so we can use infected_count in a for loop.
        infected_count = int(round(simulation.infection_percent*simulation.population_size, 0))

        #Infect the correct number of people
        for i in range(infected_count):
            #Infect the ith person in the population attribute
            self.population[i].is_infected = True
            self.population[i].days_infected = 1

        #Shuffle the population list so we spread the infection out randomly
        random.shuffle(self.population)


    def spread_infection(self, simulation):
        """Spread the infection to all adjacent people in the list population."""
        for i in range(len(self.population)):
            #ith person is ALIVE, see if they should be infected.
            #Don't bother infecting a dead person, they are infected and dead.
            #Check to see if adjacent Persons are infected
            if self.population[i].is_dead == False:
                #i is the first person in the list, can only check to the right [i+1].
                if i == 0:
                    if self.population[i+1].is_infected:
                        self.population[i].infect(simulation)
                #i is in the middle of the list, can check to the left [i-1] and right [i+1].
                elif i < len(self.population)-1:
                    if self.population[i-1].is_infected or self.population[i+1].is_infected:
                        self.population[i].infect(simulation)
                #i is the last person in the list, can only check to the left [i-1].
                elif i == len(self.population)-1:
                    if self.population[i-1].is_infected:
                        self.population[i].infect(simulation)


    def update(self, simulation):
        """Update the whole population by updating each individual person in the population."""
        simulation.day_number += 1

        #Call the update method for all person instances in the population
        for person in self.population:
            person.update(simulation)


    def display_statistics(self, simulation):
        """Display the current statistics of a population."""
        #Initialize values
        total_infected_count = 0
        total_death_count = 0

        #Loop through whole population
        for person in self.population:
            #Person is infected
            if person.is_infected:
                total_infected_count += 1
                #Person is dead
                if person.is_dead:
                    total_death_count += 1

        #Calculate percentage of population that is infected and dead
        infected_percent = round(100*(total_infected_count/simulation.population_size), 4)
        death_percent = round(100*(total_death_count/simulation.population_size), 4)

        #Statistics summary
        print("\n-----Day # " + str(simulation.day_number) + "-----")
        print("Percentage of Population Infected: " + str(infected_percent) + "%")
        print("Percentage of Population Dead: " + str(death_percent) + "%")
        print("Total People Infected: " + str(total_infected_count) + " / " + str(simulation.population_size))
        print("Total Deaths: " + str(total_death_count) + " / " + str(simulation.population_size))


    def graphics(self):
        """A graphical representation for a population. O is healthy, I infected, X dead."""
        status = [] #A list to hold all X, I, and O to represent the status of each person

        for person in self.population:
            #Person is dead, X
            if person.is_dead:
                char = 'X'
            #Person is alive, are they infected or healthy?
            else:
                #Person is infected, I
                if person.is_infected:
                    char = 'I'
                #Person is healthy, O
                else:
                    char = 'O'

            status.append(char)

        #Print out all status characters separated by a -
        for letter in status:
            print(letter, end='-')


#The main code
#Create a simulation and population object
sim = Simulation()
pop = Population(sim)

#Set the initial infection conditions of the population
pop.initial_infection(sim)
pop.display_statistics(sim)
pop.graphics()
input("\nPress enter to begin the simulation")

#Run the simulation
for i in range(1, sim.sim_days):
    #For a single day, spread the infection, update the population, display statistics and graphics
    pop.spread_infection(sim)
    pop.update(sim)
    pop.display_statistics(sim)
    pop.graphics()

    #If it is not the last day of the simulation, pause the program
    if i != sim.sim_days - 1:
        input("\nPress enter to advance to the next day.")
```