```
class News:
    def __init__(self, voteChange):
        self.voteChange = voteChange

    def getVoteChange(self):
        return self.voteChange


class Voter:
    def __init__(self, voteProb, readProb):
        self.willVote = voteProb
        self.readsNews = readProb

    def update(self, newsFeed):
        for news in newsFeed:
            if random.random() < self.readsNews:
                self.willVote += news.getVoteChange()
                self.willVote = min(self.willVote, 100)
                self.willVote = max(self.willVote, 0)
        return self.willVote


def simpleCampaign(population_size, campaignLength, newsPerDay):
    population = [Voter(50, 0.1)] * population_size
    avg_voting_will = [50]
    for step in range(campaignLength):
        stepNews = list()
        for news in range(newsPerDay):
            stepNews.append(News(random.uniform(-0.1, 0.1)))

        votingWill = list()
        for person in random.choices(population, k=150):
            votingWill.append(person.update(stepNews))

        avg_voting_will.append(sum(votingWill) / len(votingWill))

    plt.plot(avg_voting_will)
    plt.show()


simpleCampaign(1000, 200, 50)
```

```
class Passenger:
    def __init__(self, no, tolerance):
        """
        no - which bus line does the passenger want to take
        tolerance - maximal density of a bus on which the passenger will get;
                    the value should be between 0.0 and 1.0
        """
        self.no = no
        self.tolerance = tolerance

    def getNo(self):
        return self.no

    def getTolerance(self):
        return self.tolerance


class Bus:
    def __init__(self, no, capacity, passengerNo):
        """
        no - the line number of the bus
        capacity - maximum number of passengers on the bus
        passengerNo - number of passengers on the bus at the moment of arrival
        """
        self.no = no
        self.capacity = capacity
        self.passengerNo = passengerNo

    def add_passenger(self, psg):
        """
        represents a passenger getting on the bus, if there is place for them
        otherwise raises ValueError
        """
        if self.passengerNo < self.capacity:
            self.passengerNo += 1
        else:
            raise ValueError("Bus is full")

    def getDensity(self):
        """
        calculates the density to which the bus is full
        returns a float between 0.0 and 1.0
        """
        return self.passengerNo / self.capacity

    def getNo(self):
        return self.no
```

(kód pokračuje na druhej strane)

```
class BusStop:
    def __init__(self, psg_list):
        """
        psg_list - passengers waiting for a bus at the given stop
        """
        self.passengers = psg_list

    def add_passengers(self, new_psgs):
        """
        represents the arrival of new passengers (new_psg) to the bus stop
        """
        self.passengers += new_psgs

    def bus_arrival(self, bus):
        """
        represents the arrival of a bus at the bus stop
        for each passenger, if the passenger wants to get on the given bus line
        and the bus is not too full for their liking, it moves the passenger to
        the bus and clears them from the stop
        """
        for psg in self.passengers[:]:
            if psg.getNo() == bus.getNo() and psg.getTolerance() >= bus.getDensity():
                try:
                    bus.add_passenger(psg)
                    self.passengers.remove(psg)
                except ValueError:
                    pass
        return len(self.passengers)


def testStopA(psgCount, bus_lines, steps):
    passengers = list()
    for psg in range(psgCount):
        passengers.append(Passenger(random.choice(bus_lines), random.randint(1, 100)))
    stop = BusStop(passengers)

    waiting = [len(passengers)]
    for step in range(steps):
        if not step % 3:
            bus = Bus(random.choice(bus_lines), 40, random.randint(10, 40))
            waiting.append(stop.bus_arrival(bus))
        if not step % 5:
            new_psgs = list()
            for x in range(3):
                new_psgs.append(
                    Passenger(random.choice(bus_lines), random.randint(1, 100)))
            stop.add_passengers(new_psgs)

    plt.plot(waiting)
    plt.show()


testStopA(20, ['10', '17', '34'], 50)
```

```
class Passenger:
    def __init__(self, no, tolerance):
        """
        no - which bus line does the passenger want to take
        tolerance - maximal density of a bus on which the passenger will get;
                    the value should be between 0.0 and 1.0
        """
        self.no = no
        self.tolerance = tolerance

    def getNo(self):
        return self.no

    def getTolerance(self):
        return self.tolerance


class Bus:
    def __init__(self, no, capacity, passengerNo):
        """
        no - the line number of the bus
        capacity - maximum number of passengers on the bus
        passengerNo - number of passengers on the bus at the moment of arrival
        """
        self.no = no
        self.capacity = capacity
        self.passengerNo = passengerNo

    def add_passenger(self, psg):
        """
        represents a passenger getting on the bus, if there is place for them
        otherwise raises ValueError
        """
        if self.passengerNo < self.capacity:
            self.passengerNo += 1
        else:
            raise ValueError("Bus is full")

    def getDensity(self):
        """
        calculates the density to which the bus is full
        returns a float between 0.0 and 1.0
        """
        return self.passengerNo / self.capacity

    def getNo(self):
        return self.no
```

(kód pokračuje na druhej strane)

```
class BusStop:
    def __init__(self, psg_list):
        """
        psg_list - passengers waiting for a bus at the given stop
        """
        self.passengers = psg_list

    def add_passengers(self, new_psgs):
        """
        represents the arrival of new passengers (new_psg) to the bus stop
        """
        self.passengers += new_psgs

    def bus_arrival(self, bus):
        """
        represents the arrival of a bus at the bus stop
        for each passenger, if the passenger wants to get on the given bus line
        and the bus is not too full for their liking, it moves the passenger to
        the bus and clears them from the stop
        """
        for psg in self.passengers[:]:
            if psg.getNo() == bus.getNo() and psg.getTolerance() >= bus.getDensity():
                try:
                    bus.add_passenger(psg)
                    self.passengers.remove(psg)
                except ValueError:
                    pass
        return len(self.passengers)


def testStopB(psgCount, bus_lines, steps):
    passengers = list()
    for psg in range(psgCount):
        passengers.append(Passenger(random.choice(bus_lines), random.random()))
    stop = BusStop(passengers)

    waiting = [len(passengers)]
    for step in range(steps):
        if not step % 3:
            bus = Bus(random.choice(bus_lines), 40, random.random())
            waiting.append(stop.bus_arrival(bus))
        if not step % 5:
            new_psgs = list()
            for x in range(3):
                new_psgs.append(
                    Passenger(random.choice(bus_lines), random.random()))
            stop.add_passengers(new_psgs)

    plt.plot(waiting)
    plt.show()


testStopB(20, ['10', '17', '34'], 50)
```