Task 2 (30%). Bloom Join

In this exercise, you will implement Bloom join in a simulated distributed setting.

Assume a distributed scenario with NodeA containing relation R and NodeB containing relation S with the following schema:

```
R(a: int, b: int)
S(b: int, c: int)
```

We would like to execute the following query using Bloom join:

```
SELECT a, b, c
FROM R, S
WHERE R.b = S.b
```

The query is processed as follows. First, NodeA will send a bloom filter of its join column (the column b of relation R) to NodeB. Then, NodeB will use this bloom filter in order to filter relevant S tuples and send them to NodeA. Finally NodeA uses these S tuples to perform a local join with R.

Questions:

Your task is to implement the `run` methods of `NodeA` and `NodeB` classes. The `run` method of `NodeA` is responsible for creating the Bloom filter and sending it to `NodeB`, as well as processing performing the local join as soon as `NodeB` has sent the relevant tuples. The `run` method of `NodeB` is responsible for using the Bloom filter sent by `NodeB` and sending the relevant tuples to `NodeA`.

In order to send an object from a Node to another, you have to use the `send` method. NodeA uses this method for sending the bloom filter. You should use `hashFunction` as the hash function for constructing the bloom filter (here, you will use only 1 hash function). Furthermore, we suggest using `BitSet` as the bit vector implementation. However, you can come up with your own data structure for a bit vector.
NodeB uses the `send` method for sending the filtered S tuples (Note that the corresponding class for the relation S implements `Serializable`).

The `getResult` method of `NodeA` is finally invoked to retrieve the join result.

The interface is given to you in `BloomJoin.java`. We will test automatically, so please do not modify the interface, as any changes to the provided interface might cause the automatic tests to fail, which will have an impact on your grade. Specifically, you are only allowed to add methods and fields in NodeA and NodeB, without changing the already provided signatures. This file contains a simple test to demonstrate how your implementation is going to be tested.

Deliverables:

- BloomJoin.java