Esercizio HASH

```
package hash;
import java.io.*;

public class Hash
{
	public final int size = 10;
	private int[] keys = new int[10];

	public Hash()
	{
		int i = 0;
		for(i=0;i<size;i++)
		{
			keys[i] = -1;
		}

		int[] tmp = (int[]) recoverStatus("myhash");
		if(tmp != null)
		{
			keys = tmp;
		}

		System.out.println(this);
	}

	public int put(int key)
	{
		System.out.println("BEFORE: " + toString());
		int tmp = hash(key);

		while(keys[tmp] != -1 && keys[tmp] != key)
		{
			tmp = rehash(tmp);
		}
		int retval = keys[tmp];
		keys[tmp] = key;
		System.out.println("AFTER: " + toString());
		return retval;
	}

	public int get(int key)
	{
		int retval = -1;
		// to be implemented
		return retval;
	}

	public int remove(int key)
	{
		System.out.println("BEFORE: " + toString());
		int tmp = hash(key);
		System.out.println(tmp);

		while(keys[tmp] != -1 && keys[tmp] != key)
```

```
            {
                  tmp = rehash(tmp);
                  System.out.println(tmp);
            }

            if(keys[tmp] == -1)
            {
                  System.out.println(key + " non trovato");
                  return -1;
            }
            System.out.println("Compatta");

            int retval = tmp;
            keys[tmp] = -1;
            int next = rehash(tmp);
            while(keys[next] != -1)
            {
                  boolean flag= false;
                  System.out.println("keys[next] <" + keys[next] + ">
hash(keys[next]) <" + hash(keys[next]) + "> tmp <" + tmp + "> next <" + next +
">");

                  if(hash(keys[next]) > next)
                  {
                        System.out.println("hash chiave compattanda > sua
posizione");
                        flag = true;
                  }

                  if(hash(keys[next]) < tmp && next > tmp)
                  {
                        System.out.println("hash chiave compattanda < posizione
svuotata e posizione svuotanda > posizione svuotata");
                        flag = true;
                  }

                  if(hash(keys[next]) == tmp)
                  {
                        System.out.println("hash chiave compattanda = posizione
svuotata");
                        flag = true;
                  }

                  if(flag)
                  {
                        doCompatta(tmp, next);
                        System.out.println(toString());
                        tmp = next;
                  }

                  next = rehash(next);
            }
            System.out.println("AFTER: " + toString());
            return -1;
      }

      private void doCompatta(int tmp, int next)
```

```
{
	System.out.println("DO COMPATTA");
	keys[tmp] = keys[next];
	keys[next] = -1;
}

public String toString()
{
	String retval = new String("");
	int i = 0;
	for(i=0;i<size;i++)
	{
		retval=retval.concat(String.valueOf(keys[i]));
		retval = retval.concat("; ");
	}
	return retval;
}

public static void main(String[] argv) throws Exception
{
	String line = null;
	Hash tab = new Hash();

	do
	{
		System.out.print("Command? > ");
		line = readInputLine();
		if(line.equals("put"))
		{
			System.out.print("key? > ");
			String k = readInputLine();
			int key = (new Integer(k)).intValue();
			tab.put(key);
		}
		else if(line.equals("remove"))
		{
			System.out.print("key? > ");
			String k = readInputLine();
			int key = (new Integer(k)).intValue();
			tab.remove(key);
		}
		else
		{
			System.out.println("DOH!!!! <" + line + ">");
		}
	}
	while(!line.equals(""));

	storeStatus(tab.keys, "myhash");
}

private int hash(int k)
{
	return k%size;
}

private int rehash(int k)
```

```
{
      return (k+1)%size;
}


private static BufferedReader br = null;

public static String readInputLine()
{
      String retval = null;
      try
      {
            retval = br.readLine();
      }
      catch(IOException ioe)
      {
            ioe.printStackTrace();
      }
      return retval;
}

public static boolean storeStatus(Object o, String nomeFile)
{
      boolean retval = false;
      try
      {
            FileOutputStream fos = new FileOutputStream(nomeFile);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(o);
            oos.close();
            fos.close();
            retval = true;
      }
      catch(IOException ioe)
      {
            ioe.printStackTrace();
      }
      return retval;
}

public static Object recoverStatus(String nomeFile)
{
      Object retval = null;
      try
      {
            FileInputStream fis = new FileInputStream(nomeFile);
            ObjectInputStream ois = new ObjectInputStream(fis);
            retval = ois.readObject();
            ois.close();
            fis.close();
      }
      catch(ClassNotFoundException cnfe)
      {
            cnfe.printStackTrace();
      }
      catch(IOException ioe)
      {
```

```
                ioe.printStackTrace();
            }
            return retval;
        }

        static
        {
            br = new BufferedReader(new InputStreamReader(System.in));
        }
}
```