Description
Create Music School Management System (MSMS) in Java. MSMS is a system used to manage music lessons, instruments, and students at a music school. It has the following features:

1. Storage for musical instruments
2. Ability to add instruments to the school's inventory
3. Ability to remove instruments from the inventory
4. Ability to print the instrument inventory information on the console

MSMS Structure
We will need the following classes for the software:

1. Instrument – Represents a musical instrument.
2. MSMS – Music School Management System.
3. Student – Represents a student at the music school.
4. MusicSchoolTester – The tester class to test our management system.

Class Instrument
The class Instrument should have fields for the instrument name and instrument type. This class can be implemented as follows:

```
package musicschool;

public class Instrument {
    private String instrumentName, instrumentType;

    public String getInstrumentName() {
        return instrumentName;
    }

    public void setInstrumentName(String instrumentName) {
        this.instrumentName = instrumentName;
    }

    public String getInstrumentType() {
        return instrumentType;
    }

    public void setInstrumentType(String instrumentType) {
        this.instrumentType = instrumentType;
    }

    @Override
    public String toString() {
        return "Instrument: " + instrumentName + " (Type: " + instrumentType + ")";
    }
}
```

Note the setters, getters, and the toString() method implementation for the Instrument class.

Class MSMS
The Music School Management System should have an internal structure for storing instruments. It should have methods for adding new instruments and removing existing ones, as well as the ability to print the entire instrument inventory when needed. The class can be implemented as follows:

```
package musicschool;

import java.util.ArrayList;
import java.util.List;
```

```
public class MSMS {
    private List<Instrument> instrumentStorage = new ArrayList<>();

    public void addInstrument(Instrument instrument) {
        instrumentStorage.add(instrument);
    }

    public boolean removeInstrument(Instrument instrument) {
        boolean removed = false;
        for (int i = 0; i < instrumentStorage.size(); i++) {
            Instrument inst = instrumentStorage.get(i);
            if (inst.getInstrumentName().equals(instrument.getInstrumentName()) &&
                inst.getInstrumentType().equals(instrument.getInstrumentType())) {
                instrumentStorage.remove(i);
                removed = true;
                break;
            }
        }
        return removed;
    }

    public void printInstrumentStorage() {
        if (instrumentStorage.isEmpty()) {
            System.out.println("The instrument storage is empty.");
        } else {
            for (Instrument instrument : instrumentStorage) {
                System.out.println(instrument.toString());
            }
        }
    }
}
```

Pay attention to the usage of the ArrayList class, for loops for lists, the break clause, and the object comparison.

Class Student
The Student class represents a student at the music school. It has fields for the student's name, surname, and personal number.

```
package musicschool;

public class Student {
    private String name, surname, personalNumber;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }
```

```
    public String getPersonalNumber() {
        return personalNumber;
    }

    public void setPersonalNumber(String personalNumber) {
        this.personalNumber = personalNumber;
    }

    @Override
    public String toString() {
        return "Student: " + name + " " + surname + " (Personal Number: " + personalNumber + ")";
    }
}
```

Note the setters, getters, and the toString() method implementation for the Student class.

MusicSchoolTester Class
Now let's test our management system. First, create some instruments and students. Then create an instance of MSMS and add the instruments to the school's inventory. Try to remove some instruments as well.

```
package musicschool;

public class MusicSchoolTester {
    public static void main(String[] args) {
        Student giorgi = new Student();
        giorgi.setName("Giorgi");
        giorgi.setSurname("Sakvarelidze");
        giorgi.setPersonalNumber("01234567890");

        Student nino = new Student();
        nino.setName("Nino");
        nino.setSurname("Chkheidze");
        nino.setPersonalNumber("98765432109");

        Instrument panduri = new Instrument();
        panduri.setInstrumentName("Panduri");
        panduri.setInstrumentType("Lute");

        Instrument salamuri = new Instrument();
        salamuri.setInstrumentName("Salamuri");
        salamuri.setInstrumentType("Wind Instrument");

        MSMS msms = new MSMS();
        msms.addInstrument(panduri);
        msms.addInstrument(salamuri);
        msms.addInstrument(salamuri);
        msms.removeInstrument(salamuri);

        msms.printInstrumentStorage();
    }
}
```

We print the state of the instrument storage to check if all the methods are working properly.

In this implementation:
- The Student instances use Georgian names and surnames (Giorgi Sakvarelidze and Nino Chkheidze).

- The Instrument instances include traditional Georgian musical instruments like Panduri (a type of lute) and Salamuri (a wind instrument).
- The toString() method of the Instrument class provides a brief description of the instrument and its type.

The MusicSchoolTester class creates instances of Student and Instrument, adds instruments to the MSMS instance, removes one instrument, and finally prints the updated instrument storage.