# ITEC 3150 Final Exam

# Coding Problems

## For all problems, the starter files are provided in the ITEC3150FInalExam.zip file. Once you are done with all the coding problems, you can simply zip your resulting IntelliJ project using *File -> Export -> Project to Zip File ….*

If you have difficulty opening the provided starter project, you may want to open a new project file in IntelliJ, confirm that it runs and then copy the needed base files from the starter project into your project.

## Problem 1

In the main method of *HashSetExample* (provided in the starter project), two HashSet's of type String are defined named *softdev* and *syssec*.  Create a new HashSet named *common* and populate it with only the String values that are contained in both *softdev* and in *syssec*.  Next, create an additional HashSet named sdExclusive that contains those elements in softdev that are not in syssec. You may use the HashSet implementation provided in the Java library.

## Output

Common Courses

ITEC 2140

ITEC 2150

ITEC 3150

ITEC 2201

Courses Exclusive to Software Development

ITEC 3870

ITEC 4860

ITEC 3860

ITEC 4260

### Rubric

15 - Common courses constructed and reported properly

10 - Exclusive courses constructed and reported properly

### Submission

Include your modified HashSetExample.java file in your submitted IntelliJ project zip.

## Problem 2

In the main method of HashMapUser.java that is provided, create a HashMap. You may use the java.util.HashMap implementation rather than creating your own.  Add the names in the *Name* column (see following table) as the keys to the map.  The value associated with each name should be a UUID, as follows:

|  |  |
| --- | --- |
| **Name** | **UUID** |
| Riker | ad2526b1-9e83-454f-bbb4-446b68022e93 |
| Raffi | 02004051-016b-4a6c-9140-534967effc11 |
| Jean-Luc | d0a06005-b592-4750-bdfe-f7dc2d38d544 |
| Deanna | c1d4afe2-8dd0-425d-90bb-8a0244ed5774 |
| Geordi | 1469f88d-33ba-4e5f-afe0-a8dfd3bfeaa1 |
| Q | 6f8c8e52-5d94-4a3a-96e0-912fc77aeec8 |
| Locutus | e1f73186-1b64-4621-988b-6cb38df7fc62 |
| Beverly | 9cfb98d0-cce8-41fd-bb40-0fcb66e2f5ac |
| Seven | 02004051-016b-4a6c-9140-534967effc7c |

### Hint

Use UUID.fromsString(…) to instantiate the UUIDs, like:

UUID uuid1 = UUID.fromString("ad2526b1-9e83-454f-bbb4-446b68022e93");

### There is nothing special or mysterious about UUID. UUID is a class provided with the Java SDK. A UUID is like a serial number, it is a code that can be used to uniquely identify other objects like Persons, Employees, Products, etc. If I were creating an ArrayList<UUID>, I would enter:

ArrayList<UUID> uuids = new ArrayList<>();

uuids.add(UUID.fromString(("ad2526b1-9e83-454f-bbb4-446b68022e93"));

uuids.add(UUID.fromString(("02004051-016b-4a6c-9140-534967effc11"));

uuids.add(UUID.fromString(("d0a06005-b592-4750-bdfe-f7dc2d38d544"));

You aren’t going to create an ArrayList of UUIDs, but you should the pattern when creating your HashMap<String, UUID>.

### Output

Key: Locutus, Value: e1f73186-1b64-4621-988b-6cb38df7fc62

Key: Q, Value: 6f8c8e52-5d94-4a3a-96e0-912fc77aeec8

Key: Jean-Luc, Value: d0a06005-b592-4750-bdfe-f7dc2d38d544

Key: Riker, Value: ad2526b1-9e83-454f-bbb4-446b68022e93

Key: Deanna, Value: c1d4afe2-8dd0-425d-90bb-8a0244ed5774

Key: Raffi, Value: 02004051-016b-4a6c-9140-534967effc11

Key: Seven, Value: 02004051-016b-4a6c-9140-534967effc7f

Key: Geordi, Value: 1469f88d-33ba-4e5f-afe0-a8dfd3bfeaa1

Key: Beverly, Value: 9cfb98d0-cce8-41fd-bb40-0fcb66e2f5ac

Map contains 9 entries.

### Rubric

10 – HashMap Instantiated Properly, With Generics

5 – Key, Value pairs add to map as described

5 – Map reported properly

5 – Size of Map reported properly

### Submission

Include your modified HashMapUser.java file in your submitted IntelliJ project zip.

## Problem 3

Starting with BST2.java, create a new Java file named TreeTester.java with a main method.  You do not need to modify BST2.java for this question - simply use it.

In the main method, create an instance of BST2 that contains nine UUID values. Use these String values to instantiate the 9 UUID objects:

|  |
| --- |
| ad2526b1-9e83-454f-bbb4-446b68022e93 |
| 02004051-016b-4a6c-9140-534967effc11 |
| d0a06005-b592-4750-bdfe-f7dc2d38d544 |
| c1d4afe2-8dd0-425d-90bb-8a0244ed5774 |
| 1469f88d-33ba-4e5f-afe0-a8dfd3bfeaa1 |
| 6f8c8e52-5d94-4a3a-96e0-912fc77aeec8 |
| e1f73186-1b64-4621-988b-6cb38df7fc62 |
| 9cfb98d0-cce8-41fd-bb40-0fcb66e2f5ac |
| 02004051-016b-4a6c-9140-534967effc7c |

Insert these UUIDs to the BST2 tree instance. Again, instantiate the UUID objects using the fromString() static factory method:

UUID uuid1 = UUID.fromString("ad2526b1-9e83-454f-bbb4-446b68022e93");

Add code to your main() that prints the values in the tree with the inorder() method.

### Output

9cfb98d0-cce8-41fd-bb40-0fcb66e2f5ac

ad2526b1-9e83-454f-bbb4-446b68022e93

c1d4afe2-8dd0-425d-90bb-8a0244ed5774

d0a06005-b592-4750-bdfe-f7dc2d38d544

e1f73186-1b64-4621-988b-6cb38df7fc62

02004051-016b-4a6c-9140-534967effc11

02004051-016b-4a6c-9140-534967effc7c

1469f88d-33ba-4e5f-afe0-a8dfd3bfeaa1

6f8c8e52-5d94-4a3a-96e0-912fc77aeec8

Tree has 9 nodes.

Note that it is acceptable if all UUID strings print on a single line, space delimited, like this:

9cfb98d0-cce8-41fd-bb40-0fcb66e2f5ac ad2526b1-9e83-454f… … …

### Rubric

5 – BST2<UUID> tree instantiated correctly

5 - UUID objects instantiated correctly

5 - UUID inserted into tree correctly

5 - Tree is printed in order, produced correctly using method(s) in BST2

5 - Tree size reported correctly, using method(s) in BST2

### Submission

Include your modified TreeTester.java file in your submitted IntelliJ project zip.