**Assignment 3 Version Control logs**

**Trent Johnston – 10023544**

commit b5895164138d7c4c7e1e6dbac8c4b3cd8a2d793d

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 14:52:42

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 14:52:42

Child: 4e49ce1f1dcfe8523ebc81d6801f4002f68b1c31 (Initial development of Object Creation)

Branches: master, origin/master

Initial Creation

diff --git a/src/Receiver.java b/src/Receiver.java

new file mode 100644

index 0000000..a976cfc

--- /dev/null

+++ b/src/Receiver.java

@@ -0,0 +1,6 @@

+import org.jdom2.\*;

+

+

+public class Receiver {

+

+}

diff --git a/src/Sender.java b/src/Sender.java

new file mode 100644

index 0000000..42a8204

--- /dev/null

+++ b/src/Sender.java

@@ -0,0 +1,5 @@

+import org.jdom2.\*;

+

+public class Sender {

+

+}

commit 4e49ce1f1dcfe8523ebc81d6801f4002f68b1c31

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 15:56:57

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 15:56:57

Parent: b5895164138d7c4c7e1e6dbac8c4b3cd8a2d793d (Initial Creation)

Child: 193814e51b8405da362b24e74f0216212d328894 (Broke Sender into ObjectCreator and Serializer.)

Branches: master, origin/master

Initial development of Object Creation

diff --git a/src/Receiver.java b/src/Receiver.java

index a976cfc..d1f8d78 100644

--- a/src/Receiver.java

+++ b/src/Receiver.java

@@ -2,5 +2,15 @@

public class Receiver {

+ Object obj = null;

+

+ public void deserializer() {

+

+ }

+

+ public static void main() {

+

+ }

+

}

diff --git a/src/Sender.java b/src/Sender.java

index 42a8204..78e7ba2 100644

--- a/src/Sender.java

+++ b/src/Sender.java

@@ -1,5 +1,85 @@

import org.jdom2.\*;

+import java.util.Collections.\*;

+import java.util.ArrayList;

+import java.util.List;

+import java.util.Scanner;

public class Sender {

+ public List<UserObject> objList = new ArrayList<UserObject>();

+ static int nextID = 0;

+ static Scanner s = new Scanner(System.in);

+

+ public static void objCreator() {

+ boolean fin = false;

+ System.out.println("Select an option: \n"

+ + " OP - Single object with primitives \n"

+ + " OR - Single object with object references"

+ + " AP - An object containing an Array of primitives \n"

+ + " AO - An object containing an Array of objects "

+ + " C - collection class");

+ while (!fin) {

+ String line = s.nextLine();

+ if (line.equals("OP")) {

+ fin = true;

+ createOP();

+ }

+ else if (line.equals("OR")) {

+ fin = true;

+ createOR();

+ }

+ else if (line.equals("AP")) {

+ fin = true;

+ createAP();

+ }

+ else if (line.equals("AO")) {

+ fin = true;

+ createAO();

+ }

+ else if (line.equals("C")) {

+ fin = true;

+ createCollection();

+ }

+ else {

+ System.out.println("That was not a valid entry");

+ }

+ }

+ }

+

+ private static void createOP() {

+ System.out.println("Give the class name:");

+ String cls = s.nextLine();

+ UserObject obj = new UserObject(cls, nextID, "OP");

+

+ }

+ private static void createOR() {

+ // TODO Auto-generated method stub

+

+ }

+

+ private static void createAP() {

+ // TODO Auto-generated method stub

+

+ }

+

+ private static void createAO() {

+ // TODO Auto-generated method stub

+

+ }

+ private static void createCollection() {

+ // TODO Auto-generated method stub

+

+ }

+

+ public static void serializer() {

+

+ }

+

+ public static void main(String[] args) {

+ objCreator();

+ serializer();

+

+ //send over socket

+

+ }

}

diff --git a/src/UserObject.java b/src/UserObject.java

new file mode 100644

index 0000000..7b80ff3

--- /dev/null

+++ b/src/UserObject.java

@@ -0,0 +1,16 @@

+import java.util.List;

+import java.util.ArrayList;

+

+public class UserObject {

+ String cls;

+ int id;

+ String objType;

+ ArrayList<Object> fields = new ArrayList<Object>();

+

+ public UserObject(String cls, int id, String objType) {

+ this.cls = cls;

+ this.id = id;

+ this.objType = objType;

+

+ }

+}

commit 193814e51b8405da362b24e74f0216212d328894

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 16:03:33

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 16:03:33

Parent: 4e49ce1f1dcfe8523ebc81d6801f4002f68b1c31 (Initial development of Object Creation)

Child: eec98bc453de2f0f6f6317a1671852b6ae500ae3 (Completed create Object Primitive method. )

Branches: master, origin/master

Broke Sender into ObjectCreator and Serializer.

Done to avoid the bad smell of a class doing too many things.

diff --git a/src/ObjectCreator.java b/src/ObjectCreator.java

new file mode 100644

index 0000000..8530896

--- /dev/null

+++ b/src/ObjectCreator.java

@@ -0,0 +1,73 @@

+import java.util.Collections.\*;

+import java.util.ArrayList;

+import java.util.List;

+import java.util.Scanner;

+

+public class ObjectCreator {

+

+ public List<UserObject> objList = new ArrayList<UserObject>();

+ int nextID = 0;

+ Scanner s = new Scanner(System.in);

+

+ public void objCreator() {

+ boolean fin = false;

+ System.out.println("Select an option: \n"

+ + " OP - Single object with primitives \n"

+ + " OR - Single object with object references"

+ + " AP - An object containing an Array of primitives \n"

+ + " AO - An object containing an Array of objects "

+ + " C - collection class");

+ while (!fin) {

+ String line = s.nextLine();

+ if (line.equals("OP")) {

+ fin = true;

+ createOP();

+ }

+ else if (line.equals("OR")) {

+ fin = true;

+ createOR();

+ }

+ else if (line.equals("AP")) {

+ fin = true;

+ createAP();

+ }

+ else if (line.equals("AO")) {

+ fin = true;

+ createAO();

+ }

+ else if (line.equals("C")) {

+ fin = true;

+ createCollection();

+ }

+ else {

+ System.out.println("That was not a valid entry");

+ }

+ }

+ }

+

+ private void createOP() {

+ System.out.println("Give the class name:");

+ String cls = s.nextLine();

+ UserObject obj = new UserObject(cls, nextID, "OP");

+

+ }

+ private void createOR() {

+ // TODO Auto-generated method stub

+

+ }

+

+ private void createAP() {

+ // TODO Auto-generated method stub

+

+ }

+

+ private void createAO() {

+ // TODO Auto-generated method stub

+

+ }

+

+ private void createCollection() {

+ // TODO Auto-generated method stub

+

+ }

+}

diff --git a/src/Sender.java b/src/Sender.java

index 78e7ba2..5673427 100644

--- a/src/Sender.java

+++ b/src/Sender.java

@@ -1,83 +1,11 @@

-import org.jdom2.\*;

-import java.util.Collections.\*;

-import java.util.ArrayList;

-import java.util.List;

-import java.util.Scanner;

-

public class Sender {

- public List<UserObject> objList = new ArrayList<UserObject>();

- static int nextID = 0;

- static Scanner s = new Scanner(System.in);

-

- public static void objCreator() {

- boolean fin = false;

- System.out.println("Select an option: \n"

- + " OP - Single object with primitives \n"

- + " OR - Single object with object references"

- + " AP - An object containing an Array of primitives \n"

- + " AO - An object containing an Array of objects "

- + " C - collection class");

- while (!fin) {

- String line = s.nextLine();

- if (line.equals("OP")) {

- fin = true;

- createOP();

- }

- else if (line.equals("OR")) {

- fin = true;

- createOR();

- }

- else if (line.equals("AP")) {

- fin = true;

- createAP();

- }

- else if (line.equals("AO")) {

- fin = true;

- createAO();

- }

- else if (line.equals("C")) {

- fin = true;

- createCollection();

- }

- else {

- System.out.println("That was not a valid entry");

- }

- }

- }

-

- private static void createOP() {

- System.out.println("Give the class name:");

- String cls = s.nextLine();

- UserObject obj = new UserObject(cls, nextID, "OP");

-

- }

- private static void createOR() {

- // TODO Auto-generated method stub

-

- }

-

- private static void createAP() {

- // TODO Auto-generated method stub

-

- }

-

- private static void createAO() {

- // TODO Auto-generated method stub

-

- }

- private static void createCollection() {

- // TODO Auto-generated method stub

-

- }

-

- public static void serializer() {

-

- }

public static void main(String[] args) {

- objCreator();

- serializer();

+ ObjectCreator oc = new ObjectCreator();

+ oc.objCreator();

+ Serializer s = new Serializer();

+ s.serializer();

//send over socket

diff --git a/src/Serializer.java b/src/Serializer.java

new file mode 100644

index 0000000..28c8400

--- /dev/null

+++ b/src/Serializer.java

@@ -0,0 +1,9 @@

+import org.jdom2.\*;

+

+public class Serializer {

+

+

+ public void serializer() {

+

+ }

+}

commit eec98bc453de2f0f6f6317a1671852b6ae500ae3

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 16:53:26

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 16:53:26

Parent: 193814e51b8405da362b24e74f0216212d328894 (Broke Sender into ObjectCreator and Serializer.)

Child: cdc621a86e65e6f68395c511cb9180ab3b33d0e4 (Completed createOR)

Branches: master, origin/master

Completed create Object Primitive method.

Evolved UserObject and created UserField to accommodate

diff --git a/src/ObjectCreator.java b/src/ObjectCreator.java

index 8530896..2e51273 100644

--- a/src/ObjectCreator.java

+++ b/src/ObjectCreator.java

@@ -16,7 +16,7 @@

+ " OR - Single object with object references"

+ " AP - An object containing an Array of primitives \n"

+ " AO - An object containing an Array of objects "

- + " C - collection class");

+ + " C - An object containing a collection class");

while (!fin) {

String line = s.nextLine();

if (line.equals("OP")) {

@@ -45,29 +45,81 @@

}

}

- private void createOP() {

- System.out.println("Give the class name:");

+ public void createOP() {

+ System.out.println("Give the class name for new object (id# " + nextID + "):");

String cls = s.nextLine();

UserObject obj = new UserObject(cls, nextID, "OP");

-

+ int objID = nextID;

+ boolean fin = false;

+ while (!fin) {

+ System.out.println("Create primitive fields for " + cls + " (id# " + objID + ") in the form <name> <value> or '$$' to finish");

+ String line = s.nextLine();

+ String[] sLine = line.split(" ");

+ if (line.equals("$$")) {

+ fin = true;

+ }

+ else if (sLine.length != 2) {

+ System.out.println("Incorrect formatting; Form <name> <value> or '$$' to finish");

+ }

+ else if (!(isNum(sLine[1]) || isChar(sLine[1]))){

+ System.out.println("value was not a primitive");

+ }

+ else {

+ obj.fields.add(new UserField(obj.cls, sLine[0], sLine[1]));

+ }

+ }

+ prepareNextID();

}

- private void createOR() {

+ private boolean isNum(String str) {

+ try {

+ Double d = Double.parseDouble(str);

+ } catch (NumberFormatException e) {

+ return false;

+ }

+ return true;

+ }

+

+ private boolean isChar(String str) {

+ if (str.length() == 1) {

+ return true;

+ }

+ else {

+ return false;

+ }

+ }

+

+ public void createOR() {

// TODO Auto-generated method stub

}

- private void createAP() {

+ public void createAP() {

// TODO Auto-generated method stub

}

- private void createAO() {

+ public void createAO() {

// TODO Auto-generated method stub

}

- private void createCollection() {

+ public void createCollection() {

// TODO Auto-generated method stub

}

+

+ public void prepareNextID() {

+ nextID++;

+ boolean tryAgain = false;

+ while (!tryAgain) {

+ tryAgain = false;

+ for (UserObject obj : objList) {

+ if (obj.id == nextID) {

+ nextID++;

+ tryAgain = true;

+ break;

+ }

+ }

+ }

+ }

}

diff --git a/src/UserField.java b/src/UserField.java

new file mode 100644

index 0000000..79e02f9

--- /dev/null

+++ b/src/UserField.java

@@ -0,0 +1,19 @@

+

+public class UserField {

+ String declaringclass = "";

+ String name = "";

+ String value = "";

+ int reference = 0;

+

+ public UserField(String declaringclass, String name, String value) {

+ this.declaringclass = declaringclass;

+ this.name = name;

+ this.value = value;

+ }

+

+ public UserField(String declaringclass, String name, int reference) {

+ this.declaringclass = declaringclass;

+ this.name = name;

+ this.reference = reference;

+ }

+}

diff --git a/src/UserObject.java b/src/UserObject.java

index 7b80ff3..a92e6db 100644

--- a/src/UserObject.java

+++ b/src/UserObject.java

@@ -5,7 +5,7 @@

String cls;

int id;

String objType;

- ArrayList<Object> fields = new ArrayList<Object>();

+ ArrayList<UserField> fields = new ArrayList<UserField>();

public UserObject(String cls, int id, String objType) {

this.cls = cls;

commit cdc621a86e65e6f68395c511cb9180ab3b33d0e4

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 17:56:15

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 17:56:15

Parent: eec98bc453de2f0f6f6317a1671852b6ae500ae3 (Completed create Object Primitive method. )

Child: 6329a01c7bb82d40353f722bd3b3432c9b719f96 (Completed createAP (array of primitives))

Branches: master, origin/master

Completed createOR

create Object with Object References

diff --git a/src/ObjectCreator.java b/src/ObjectCreator.java

index 2e51273..e057b1d 100644

--- a/src/ObjectCreator.java

+++ b/src/ObjectCreator.java

@@ -9,50 +9,53 @@

int nextID = 0;

Scanner s = new Scanner(System.in);

- public void objCreator() {

+ public int objCreator() {

boolean fin = false;

+ int ret = 0;

System.out.println("Select an option: \n"

+ " OP - Single object with primitives \n"

- + " OR - Single object with object references"

+ + " OR - Single object with object references\n"

+ " AP - An object containing an Array of primitives \n"

- + " AO - An object containing an Array of objects "

+ + " AO - An object containing an Array of objects \n"

+ " C - An object containing a collection class");

while (!fin) {

String line = s.nextLine();

if (line.equals("OP")) {

fin = true;

- createOP();

+ ret = createOP();

}

else if (line.equals("OR")) {

fin = true;

- createOR();

+ ret = createOR();

}

else if (line.equals("AP")) {

fin = true;

- createAP();

+ ret = createAP();

}

else if (line.equals("AO")) {

fin = true;

- createAO();

+ ret = createAO();

}

else if (line.equals("C")) {

fin = true;

- createCollection();

+ ret = createCollection();

}

else {

- System.out.println("That was not a valid entry");

+ System.out.println("That was not a valid entry\n");

}

}

+ return ret;

}

- public void createOP() {

+ public int createOP() {

System.out.println("Give the class name for new object (id# " + nextID + "):");

String cls = s.nextLine();

UserObject obj = new UserObject(cls, nextID, "OP");

- int objID = nextID;

+ objList.add(obj);

+ prepareNextID();

boolean fin = false;

while (!fin) {

- System.out.println("Create primitive fields for " + cls + " (id# " + objID + ") in the form <name> <value> or '$$' to finish");

+ System.out.println("Create primitive fields for " + cls + " (id# " + obj.id + ") in the form <name> <value> or '$$' to finish");

String line = s.nextLine();

String[] sLine = line.split(" ");

if (line.equals("$$")) {

@@ -61,15 +64,27 @@

else if (sLine.length != 2) {

System.out.println("Incorrect formatting; Form <name> <value> or '$$' to finish");

}

- else if (!(isNum(sLine[1]) || isChar(sLine[1]))){

+ else if (!(isChar(sLine[1]) || isNum(sLine[1]))){

System.out.println("value was not a primitive");

+ }

+ else if (nameExists(obj, sLine[0])) {

+ System.out.println("Field name already exists");

}

else {

obj.fields.add(new UserField(obj.cls, sLine[0], sLine[1]));

}

}

- prepareNextID();

+ return obj.id;

}

+ private boolean nameExists(UserObject obj, String fName) {

+ for (UserField fields : obj.fields) {

+ if (fields.name.equals(fName)) {

+ return true;

+ }

+ }

+ return false;

+ }

+

private boolean isNum(String str) {

try {

Double d = Double.parseDouble(str);

@@ -88,24 +103,71 @@

}

}

- public void createOR() {

- // TODO Auto-generated method stub

-

+ public int createOR() {

+ System.out.println("Give the class name for new object (id# " + nextID + "):");

+ String cls = s.nextLine();

+ UserObject obj = new UserObject(cls, nextID, "OR");

+ objList.add(obj);

+ prepareNextID();

+ boolean fin = false;

+ while (!fin) {

+ System.out.println("Create object reference fields for " + cls + " (id# " + obj.id + ")\n"

+ + " Use the form <name> <refID#> for an existing object\n"

+ + " Use the form <name> N for a new object\n"

+ + " or '$$' to finish");

+ String line = s.nextLine();

+ String[] sLine = line.split(" ");

+ if (line.equals("$$")) {

+ fin = true;

+ }

+ else if (sLine.length != 2) {

+ System.out.println("Incorrect formatting; Form <name> <refID#>/N or '$$' to finish");

+ }

+ else if (nameExists(obj, sLine[0])) {

+ System.out.println("Field name already exists");

+ }

+ else if (sLine[1].equals("N")) {

+ int returnedID = objCreator();

+ obj.fields.add(new UserField(obj.cls, sLine[0], returnedID));

+ }

+ else if (!checkObjCreated(sLine[1])){

+ System.out.println("refID# " + sLine[1] + " does not exist");

+ }

+ else {

+ obj.fields.add(new UserField(obj.cls, sLine[0], Integer.parseInt(sLine[1])));

+ }

+ }

+ return obj.id;

}

- public void createAP() {

- // TODO Auto-generated method stub

-

- }

-

- public void createAO() {

- // TODO Auto-generated method stub

-

+ private boolean checkObjCreated(String string) {

+ int id = 999999999;

+ try {

+ id = Integer.parseInt(string);

+ } catch (NumberFormatException e) {

+ return false;

+ }

+ for (UserObject obj : objList) {

+ if (obj.id == id) {

+ return true;

+ }

+ }

+ return false;

}

- public void createCollection() {

+ public int createAP() {

// TODO Auto-generated method stub

-

+ return 0;

+ }

+

+ public int createAO() {

+ // TODO Auto-generated method stub

+ return 0;

+ }

+

+ public int createCollection() {

+ // TODO Auto-generated method stub

+ return 0;

}

public void prepareNextID() {

diff --git a/src/UserField.java b/src/UserField.java

index 79e02f9..b2e2126 100644

--- a/src/UserField.java

+++ b/src/UserField.java

@@ -4,16 +4,21 @@

String name = "";

String value = "";

int reference = 0;

+ boolean isPrim = false;

+ boolean isArray = false;

+ boolean isObject = false;

public UserField(String declaringclass, String name, String value) {

this.declaringclass = declaringclass;

this.name = name;

this.value = value;

+ isPrim = true;

}

public UserField(String declaringclass, String name, int reference) {

this.declaringclass = declaringclass;

this.name = name;

this.reference = reference;

+ isObject = true;

}

}

commit 6329a01c7bb82d40353f722bd3b3432c9b719f96

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 22:30:06

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 22:30:06

Parent: cdc621a86e65e6f68395c511cb9180ab3b33d0e4 (Completed createOR)

Child: df60fb2778b928c0f6381d66d37da7b1cae4bea3 (Completed createAO (create array of objects))

Branches: master, origin/master

Completed createAP (array of primitives)

diff --git a/src/ObjectCreator.java b/src/ObjectCreator.java

index e057b1d..6d2fcd3 100644

--- a/src/ObjectCreator.java

+++ b/src/ObjectCreator.java

@@ -156,8 +156,40 @@

}

public int createAP() {

- // TODO Auto-generated method stub

- return 0;

+ boolean fin = false;

+ String cls = "";

+ while (!fin) {

+ System.out.println("Give the array components type for new object (id# " + nextID + "): (B,C,D,F,I,J,S,Z)");

+ cls = s.nextLine();

+ if (cls.equals("B") || cls.equals("C") || cls.equals("D") || cls.equals("F")

+ || cls.equals("I") || cls.equals("J") || cls.equals("S") || cls.equals("Z")) {

+ fin = true;

+ }

+ else {

+ System.out.println("Invalid primitive component type");

+ }

+ }

+ UserObject obj = new UserObject("[" + cls, nextID, "AP");

+ objList.add(obj);

+ prepareNextID();

+

+ while (!fin) {

+ System.out.println("Create primitive value for [" + cls + " (id# " + obj.id + "). '$$' to finish");

+ String line = s.nextLine();

+ if (line.equals("$$")) {

+ fin = true;

+ }

+ else if (cls.equals("C") && !isChar(line)) {

+ System.out.println("Incorrect input; single characture or '$$' to finish");

+ }

+ else if (!cls.equals("C") && !isNum(line)) {

+ System.out.println("Incorrect input; number or '$$' to finish");

+ }

+ else {

+ obj.primArray.add(line);

+ }

+ }

+ return obj.id;

}

public int createAO() {

diff --git a/src/UserField.java b/src/UserField.java

index b2e2126..64807f0 100644

--- a/src/UserField.java

+++ b/src/UserField.java

@@ -5,7 +5,6 @@

String value = "";

int reference = 0;

boolean isPrim = false;

- boolean isArray = false;

boolean isObject = false;

public UserField(String declaringclass, String name, String value) {

diff --git a/src/UserObject.java b/src/UserObject.java

index a92e6db..dd4d29b 100644

--- a/src/UserObject.java

+++ b/src/UserObject.java

@@ -6,11 +6,12 @@

int id;

String objType;

ArrayList<UserField> fields = new ArrayList<UserField>();

+ ArrayList<Integer> objArray = new ArrayList<Integer>();

+ ArrayList<String> primArray = new ArrayList<String>();

public UserObject(String cls, int id, String objType) {

this.cls = cls;

this.id = id;

this.objType = objType;

-

}

}

commit df60fb2778b928c0f6381d66d37da7b1cae4bea3

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 22:40:04

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-18 22:40:04

Parent: 6329a01c7bb82d40353f722bd3b3432c9b719f96 (Completed createAP (array of primitives))

Child: f8427c6541e804e99b2651a1b586192ae3e89c9d (Completed Serializer)

Branches: master, origin/master

Completed createAO (create array of objects)

diff --git a/src/ObjectCreator.java b/src/ObjectCreator.java

index 6d2fcd3..332c073 100644

--- a/src/ObjectCreator.java

+++ b/src/ObjectCreator.java

@@ -193,8 +193,32 @@

}

public int createAO() {

- // TODO Auto-generated method stub

- return 0;

+ String cls = "[Object";

+ UserObject obj = new UserObject(cls, nextID, "AO");

+ objList.add(obj);

+ prepareNextID();

+ boolean fin = false;

+ while (!fin) {

+ System.out.println("Create object reference fields for " + cls + " (id# " + obj.id + ")\n"

+ + " Enter <refID#> for an existing object\n"

+ + " Enter N for a new object\n"

+ + " or '$$' to finish");

+ String line = s.nextLine();

+ if (line.equals("$$")) {

+ fin = true;

+ }

+ else if (line.equals("N")) {

+ int returnedID = objCreator();

+ obj.objArray.add(returnedID);

+ }

+ else if (!isNum(line) || !checkObjCreated(line)){

+ System.out.println("refID# " + line + " does not exist");

+ }

+ else {

+ obj.objArray.add(Integer.parseInt(line));

+ }

+ }

+ return obj.id;

}

public int createCollection() {

commit f8427c6541e804e99b2651a1b586192ae3e89c9d

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-20 20:38:30

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-20 20:38:30

Parent: df60fb2778b928c0f6381d66d37da7b1cae4bea3 (Completed createAO (create array of objects))

Child: 3368932792ffd45ef08aacb88152c929bc50a076 (Completed Sender (Socket finished))

Branches: master, origin/master

Completed Serializer

diff --git a/src/ObjectCreator.java b/src/ObjectCreator.java

index 332c073..b388166 100644

--- a/src/ObjectCreator.java

+++ b/src/ObjectCreator.java

@@ -1,4 +1,3 @@

-import java.util.Collections.\*;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

@@ -9,9 +8,9 @@

int nextID = 0;

Scanner s = new Scanner(System.in);

- public int objCreator() {

+ public UserObject objCreator() {

boolean fin = false;

- int ret = 0;

+ UserObject ret = null;

System.out.println("Select an option: \n"

+ " OP - Single object with primitives \n"

+ " OR - Single object with object references\n"

@@ -47,7 +46,7 @@

return ret;

}

- public int createOP() {

+ public UserObject createOP() {

System.out.println("Give the class name for new object (id# " + nextID + "):");

String cls = s.nextLine();

UserObject obj = new UserObject(cls, nextID, "OP");

@@ -74,36 +73,11 @@

obj.fields.add(new UserField(obj.cls, sLine[0], sLine[1]));

}

}

- return obj.id;

- }

- private boolean nameExists(UserObject obj, String fName) {

- for (UserField fields : obj.fields) {

- if (fields.name.equals(fName)) {

- return true;

- }

- }

- return false;

- }

-

- private boolean isNum(String str) {

- try {

- Double d = Double.parseDouble(str);

- } catch (NumberFormatException e) {

- return false;

- }

- return true;

+ return obj;

}

- private boolean isChar(String str) {

- if (str.length() == 1) {

- return true;

- }

- else {

- return false;

- }

- }

- public int createOR() {

+ public UserObject createOR() {

System.out.println("Give the class name for new object (id# " + nextID + "):");

String cls = s.nextLine();

UserObject obj = new UserObject(cls, nextID, "OR");

@@ -127,8 +101,8 @@

System.out.println("Field name already exists");

}

else if (sLine[1].equals("N")) {

- int returnedID = objCreator();

- obj.fields.add(new UserField(obj.cls, sLine[0], returnedID));

+ UserObject returnedObj = objCreator();

+ obj.fields.add(new UserField(obj.cls, sLine[0], returnedObj.id));

}

else if (!checkObjCreated(sLine[1])){

System.out.println("refID# " + sLine[1] + " does not exist");

@@ -137,25 +111,10 @@

obj.fields.add(new UserField(obj.cls, sLine[0], Integer.parseInt(sLine[1])));

}

}

- return obj.id;

- }

-

- private boolean checkObjCreated(String string) {

- int id = 999999999;

- try {

- id = Integer.parseInt(string);

- } catch (NumberFormatException e) {

- return false;

- }

- for (UserObject obj : objList) {

- if (obj.id == id) {

- return true;

- }

- }

- return false;

+ return obj;

}

- public int createAP() {

+ public UserObject createAP() {

boolean fin = false;

String cls = "";

while (!fin) {

@@ -173,7 +132,21 @@

objList.add(obj);

prepareNextID();

+ fin = false;

while (!fin) {

+ System.out.println("Set the array size for " + cls + " (id# " + obj.id + ")");

+ String line = s.nextLine();

+ if (!isInt(line)) {

+ System.out.println("That was not an integer");

+ }

+ else {

+ obj.arraySize = Integer.parseInt(line);

+ fin = true;

+ }

+ }

+ fin = false;

+

+ while (!fin && obj.arraySize > obj.primArray.size()) {

System.out.println("Create primitive value for [" + cls + " (id# " + obj.id + "). '$$' to finish");

String line = s.nextLine();

if (line.equals("$$")) {

@@ -189,12 +162,53 @@

obj.primArray.add(line);

}

}

- return obj.id;

+ return obj;

}

- public int createAO() {

+ public UserObject createAO() {

String cls = "[Object";

UserObject obj = new UserObject(cls, nextID, "AO");

+ objList.add(obj);

+ prepareNextID();

+ boolean fin = false;

+ while (!fin) {

+ System.out.println("Set the array size for " + cls + " (id# " + obj.id + ")");

+ String line = s.nextLine();

+ if (!isInt(line)) {

+ System.out.println("That was not an integer");

+ }

+ else {

+ obj.arraySize = Integer.parseInt(line);

+ fin = true;

+ }

+ }

+ fin = false;

+ while (!fin && obj.arraySize > obj.objArray.size()) {

+ System.out.println("Create object reference fields for " + cls + " (id# " + obj.id + ")\n"

+ + " Enter <refID#> for an existing object\n"

+ + " Enter N for a new object\n"

+ + " or '$$' to finish");

+ String line = s.nextLine();

+ if (line.equals("$$")) {

+ fin = true;

+ }

+ else if (line.equals("N")) {

+ UserObject returnedObj = objCreator();

+ obj.objArray.add(returnedObj);

+ }

+ else if (!isInt(line) || !checkObjCreated(line)){

+ System.out.println("refID# " + line + " does not exist");

+ }

+ else {

+ obj.objArray.add(getObjFromID(Integer.parseInt(line)));

+ }

+ }

+ return obj;

+ }

+

+ public UserObject createCollection() {

+ String cls = ArrayList.class.getName();

+ UserObject obj = new UserObject(cls, nextID, "collection");

objList.add(obj);

prepareNextID();

boolean fin = false;

@@ -208,22 +222,26 @@

fin = true;

}

else if (line.equals("N")) {

- int returnedID = objCreator();

- obj.objArray.add(returnedID);

+ UserObject returnedObj = objCreator();

+ obj.objArray.add(returnedObj);

}

- else if (!isNum(line) || !checkObjCreated(line)){

+ else if (!isInt(line) || !checkObjCreated(line)){

System.out.println("refID# " + line + " does not exist");

}

else {

- obj.objArray.add(Integer.parseInt(line));

+ obj.objArray.add(getObjFromID(Integer.parseInt(line)));

}

}

- return obj.id;

+ return obj;

}

-

- public int createCollection() {

- // TODO Auto-generated method stub

- return 0;

+

+ public UserObject getObjFromID(int id) {

+ for (UserObject obj : objList) {

+ if (obj.id == id) {

+ return obj;

+ }

+ }

+ return null;

}

public void prepareNextID() {

@@ -240,4 +258,59 @@

}

}

}

+

+ private boolean checkObjCreated(String string) {

+ int id = 999999999;

+ try {

+ id = Integer.parseInt(string);

+ } catch (NumberFormatException e) {

+ return false;

+ }

+ for (UserObject obj : objList) {

+ if (obj.id == id) {

+ return true;

+ }

+ }

+ return false;

+ }

+

+ private boolean nameExists(UserObject obj, String fName) {

+ for (UserField fields : obj.fields) {

+ if (fields.name.equals(fName)) {

+ return true;

+ }

+ }

+ return false;

+ }

+

+ private boolean isNum(String str) {

+ try {

+ Double.parseDouble(str);

+ } catch (NumberFormatException e) {

+ return false;

+ }

+ return true;

+ }

+

+ private boolean isInt(String str) {

+ try {

+ Integer.parseInt(str);

+ } catch (NumberFormatException e) {

+ return false;

+ }

+ return true;

+ }

+

+ private boolean isChar(String str) {

+ if (str.length() == 1) {

+ return true;

+ }

+ else {

+ return false;

+ }

+ }

+

+ public List<UserObject> getObjList() {

+ return objList;

+ }

}

diff --git a/src/Sender.java b/src/Sender.java

index 5673427..54cf153 100644

--- a/src/Sender.java

+++ b/src/Sender.java

@@ -1,9 +1,12 @@

+import java.util.List;

+

public class Sender {

public static void main(String[] args) {

ObjectCreator oc = new ObjectCreator();

oc.objCreator();

+ List<UserObject> objList = oc.getObjList();

Serializer s = new Serializer();

s.serializer();

diff --git a/src/Serializer.java b/src/Serializer.java

index 28c8400..e6c76c6 100644

--- a/src/Serializer.java

+++ b/src/Serializer.java

@@ -1,9 +1,82 @@

+import java.util.ArrayList;

+import java.util.List;

+

import org.jdom2.\*;

public class Serializer {

-

+ ArrayList<Integer> done = new ArrayList<Integer>();

+ Element root = new Element("serialized");

public void serializer() {

}

+

+ public org.jdom2.Document serialize(Object norm\_obj) {

+ if (!(norm\_obj instanceof List<?>)) {

+ System.out.println("cannot serialize an object that is not a UserObject");

+ System.exit(-1);

+ }

+ @SuppressWarnings("unchecked")

+ ArrayList<UserObject> objList = (ArrayList<UserObject>) norm\_obj;

+ for (UserObject obj : objList) {

+ serial(obj);

+ }

+

+ return new org.jdom2.Document(root);

+ }

+

+ public void serial(UserObject obj) {

+ Element ele = new Element("object");

+ ele.setAttribute("class", obj.cls);

+ ele.setAttribute("id", String.valueOf(obj.id));

+ done.add(obj.id);

+

+ if (obj.objType.equals("OP")) {

+ for (UserField i : obj.fields) {

+ Element e = new Element("field");

+ e.setAttribute("name", i.name);

+ e.setAttribute("declaringclass", i.declaringclass);

+ e.addContent((new Element("value")).setText(i.value));

+ ele.addContent(e);

+ }

+ }

+ else if (obj.objType.equals("OR")) {

+ for (UserField i : obj.fields) {

+ Element e = new Element("field");

+ e.setAttribute("name", i.name);

+ e.setAttribute("declaringclass", i.declaringclass);

+ e.addContent((new Element("reference")).setText(String.valueOf(i.reference)));

+ ele.addContent(e);

+ }

+ }

+ else if (obj.objType.equals("AP")) {

+ ele.setAttribute("length", String.valueOf(obj.arraySize));

+ for (int i = 0; i < obj.arraySize; i++) {

+ if (i < obj.primArray.size()) {

+ ele.addContent((new Element("value")).setText(obj.primArray.get(i)));

+ }

+ else {

+ ele.addContent((new Element("value")).setText("0"));

+ }

+ }

+ }

+ else if (obj.objType.equals("AO")) {

+ ele.setAttribute("length", String.valueOf(obj.arraySize));

+ for (int i = 0; i < obj.arraySize; i++) {

+ if (i < obj.objArray.size()) {

+ ele.addContent((new Element("reference")).setText(String.valueOf(obj.objArray.get(i).id)));

+ }

+ else {

+ ele.addContent((new Element("reference")).setText("NULL"));

+ }

+ }

+ }

+ else {

+ for (int i = 0; i < obj.arraySize; i++) {

+ ele.addContent((new Element("reference")).setText(String.valueOf(obj.objArray.get(i).id)));

+ }

+ }

+

+ root.addContent(ele);

+ }

}

diff --git a/src/UserObject.java b/src/UserObject.java

index dd4d29b..3ed008d 100644

--- a/src/UserObject.java

+++ b/src/UserObject.java

@@ -1,4 +1,3 @@

-import java.util.List;

import java.util.ArrayList;

public class UserObject {

@@ -6,8 +5,9 @@

int id;

String objType;

ArrayList<UserField> fields = new ArrayList<UserField>();

- ArrayList<Integer> objArray = new ArrayList<Integer>();

+ ArrayList<UserObject> objArray = new ArrayList<UserObject>();

ArrayList<String> primArray = new ArrayList<String>();

+ int arraySize = 0;

public UserObject(String cls, int id, String objType) {

this.cls = cls;

commit 3368932792ffd45ef08aacb88152c929bc50a076

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-20 21:30:35

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-20 21:30:35

Parent: f8427c6541e804e99b2651a1b586192ae3e89c9d (Completed Serializer)

Child: 3ba336932ea005e53ff37412a4be285930bb194a (Removed type code in UserObject)

Branches: master, origin/master

Completed Sender (Socket finished)

diff --git a/src/Receiver.java b/src/Receiver.java

index d1f8d78..7738f76 100644

--- a/src/Receiver.java

+++ b/src/Receiver.java

@@ -1,15 +1,30 @@

import org.jdom2.\*;

-

+import java.util.List;

+import java.io.BufferedReader;

+import java.io.IOException;

+import java.net.ServerSocket;

+import java.net.Socket;

+import java.net.UnknownHostException;

+import java.io.InputStreamReader;

+import java.io.PrintWriter;

public class Receiver {

- Object obj = null;

+ private static ServerSocket serverSocket;

+ private static Socket clientSocket;

+ private static int port = 4444;

+ private static PrintWriter out;

+ private static BufferedReader in;

public void deserializer() {

}

- public static void main() {

-

+ public static void main() throws IOException {

+ //send over socket

+ serverSocket = new ServerSocket(port);

+ clientSocket = serverSocket.accept();

+ out = new PrintWriter(clientSocket.getOutputStream(), true);

+ in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

}

diff --git a/src/Sender.java b/src/Sender.java

index 54cf153..29d89d0 100644

--- a/src/Sender.java

+++ b/src/Sender.java

@@ -1,16 +1,47 @@

import java.util.List;

+import org.jdom2.output.XMLOutputter;

+

+import java.io.BufferedReader;

+import java.io.IOException;

+import java.net.Socket;

+import java.net.UnknownHostException;

+import java.io.InputStreamReader;

+import java.io.PrintWriter;

+

public class Sender {

+ private static Socket clientSocket;

+ private static PrintWriter out;

+ private static BufferedReader in;

-

- public static void main(String[] args) {

+ public static void startConnection(String ip, int port) throws UnknownHostException, IOException {

+ clientSocket = new Socket(ip, port);

+ out = new PrintWriter(clientSocket.getOutputStream(), true);

+ in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

+ }

+

+ /\*public String sendMessage(String msg) throws IOException {

+ out.println(msg);

+ String resp = in.readLine();

+ return resp;

+ }\*/

+

+ public static void stopConnection() throws IOException {

+ in.close();

+ out.close();

+ clientSocket.close();

+ }

+

+ public static void main(String[] args) throws UnknownHostException, IOException {

ObjectCreator oc = new ObjectCreator();

oc.objCreator();

List<UserObject> objList = oc.getObjList();

Serializer s = new Serializer();

- s.serializer();

+ org.jdom2.Document doc = s.serialize(objList);

- //send over socket

+ startConnection("127.0.0.1", 4444);

+ out.println(new XMLOutputter().outputString(doc));

+

}

}

diff --git a/src/Serializer.java b/src/Serializer.java

index e6c76c6..569bc7c 100644

--- a/src/Serializer.java

+++ b/src/Serializer.java

@@ -6,10 +6,6 @@

public class Serializer {

ArrayList<Integer> done = new ArrayList<Integer>();

Element root = new Element("serialized");

-

- public void serializer() {

-

- }

public org.jdom2.Document serialize(Object norm\_obj) {

if (!(norm\_obj instanceof List<?>)) {

commit 3ba336932ea005e53ff37412a4be285930bb194a

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 16:25:29

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 16:25:29

Parent: 3368932792ffd45ef08aacb88152c929bc50a076 (Completed Sender (Socket finished))

Child: d870ce8902610814137a9f2a6f65dd1b83a3006e (Completed Deserializer)

Branches: master, origin/master

Removed type code in UserObject

diff --git a/src/ObjectCreator.java b/src/ObjectCreator.java

index b388166..21431f5 100644

--- a/src/ObjectCreator.java

+++ b/src/ObjectCreator.java

@@ -49,7 +49,7 @@

public UserObject createOP() {

System.out.println("Give the class name for new object (id# " + nextID + "):");

String cls = s.nextLine();

- UserObject obj = new UserObject(cls, nextID, "OP");

+ UserObject obj = new UserObject(cls, nextID);

objList.add(obj);

prepareNextID();

boolean fin = false;

@@ -80,7 +80,7 @@

public UserObject createOR() {

System.out.println("Give the class name for new object (id# " + nextID + "):");

String cls = s.nextLine();

- UserObject obj = new UserObject(cls, nextID, "OR");

+ UserObject obj = new UserObject(cls, nextID);

objList.add(obj);

prepareNextID();

boolean fin = false;

@@ -128,7 +128,7 @@

System.out.println("Invalid primitive component type");

}

}

- UserObject obj = new UserObject("[" + cls, nextID, "AP");

+ UserObject obj = new UserObject("[" + cls, nextID);

objList.add(obj);

prepareNextID();

@@ -167,7 +167,7 @@

public UserObject createAO() {

String cls = "[Object";

- UserObject obj = new UserObject(cls, nextID, "AO");

+ UserObject obj = new UserObject(cls, nextID);

objList.add(obj);

prepareNextID();

boolean fin = false;

@@ -194,13 +194,13 @@

}

else if (line.equals("N")) {

UserObject returnedObj = objCreator();

- obj.objArray.add(returnedObj);

+ obj.objArray.add(returnedObj.id);

}

else if (!isInt(line) || !checkObjCreated(line)){

System.out.println("refID# " + line + " does not exist");

}

else {

- obj.objArray.add(getObjFromID(Integer.parseInt(line)));

+ obj.objArray.add(Integer.parseInt(line));

}

}

return obj;

@@ -208,7 +208,7 @@

public UserObject createCollection() {

String cls = ArrayList.class.getName();

- UserObject obj = new UserObject(cls, nextID, "collection");

+ UserObject obj = new UserObject(cls, nextID);

objList.add(obj);

prepareNextID();

boolean fin = false;

@@ -223,13 +223,13 @@

}

else if (line.equals("N")) {

UserObject returnedObj = objCreator();

- obj.objArray.add(returnedObj);

+ obj.objArray.add(returnedObj.id);

}

else if (!isInt(line) || !checkObjCreated(line)){

System.out.println("refID# " + line + " does not exist");

}

else {

- obj.objArray.add(getObjFromID(Integer.parseInt(line)));

+ obj.objArray.add(Integer.parseInt(line));

}

}

return obj;

diff --git a/src/Serializer.java b/src/Serializer.java

index 569bc7c..eda764f 100644

--- a/src/Serializer.java

+++ b/src/Serializer.java

@@ -27,7 +27,7 @@

ele.setAttribute("id", String.valueOf(obj.id));

done.add(obj.id);

- if (obj.objType.equals("OP")) {

+ if (obj.fields.size() > 0 && obj.fields.get(0).isPrim) {

for (UserField i : obj.fields) {

Element e = new Element("field");

e.setAttribute("name", i.name);

@@ -36,7 +36,7 @@

ele.addContent(e);

}

}

- else if (obj.objType.equals("OR")) {

+ else if (obj.fields.size() > 0 && obj.fields.get(0).isObject) {

for (UserField i : obj.fields) {

Element e = new Element("field");

e.setAttribute("name", i.name);

@@ -45,7 +45,7 @@

ele.addContent(e);

}

}

- else if (obj.objType.equals("AP")) {

+ else if (obj.primArray.size() > 0 && obj.arraySize > 0) {

ele.setAttribute("length", String.valueOf(obj.arraySize));

for (int i = 0; i < obj.arraySize; i++) {

if (i < obj.primArray.size()) {

@@ -56,11 +56,11 @@

}

}

}

- else if (obj.objType.equals("AO")) {

+ else if (obj.objArray.size() > 0 && obj.arraySize > 0) {

ele.setAttribute("length", String.valueOf(obj.arraySize));

for (int i = 0; i < obj.arraySize; i++) {

if (i < obj.objArray.size()) {

- ele.addContent((new Element("reference")).setText(String.valueOf(obj.objArray.get(i).id)));

+ ele.addContent((new Element("reference")).setText(String.valueOf(obj.objArray.get(i))));

}

else {

ele.addContent((new Element("reference")).setText("NULL"));

@@ -68,8 +68,8 @@

}

}

else {

- for (int i = 0; i < obj.arraySize; i++) {

- ele.addContent((new Element("reference")).setText(String.valueOf(obj.objArray.get(i).id)));

+ for (int i = 0; i < obj.objArray.size(); i++) {

+ ele.addContent((new Element("reference")).setText(String.valueOf(obj.objArray.get(i))));

}

}

diff --git a/src/UserObject.java b/src/UserObject.java

index 3ed008d..c6126ac 100644

--- a/src/UserObject.java

+++ b/src/UserObject.java

@@ -3,15 +3,19 @@

public class UserObject {

String cls;

int id;

- String objType;

+ //String objType;

ArrayList<UserField> fields = new ArrayList<UserField>();

- ArrayList<UserObject> objArray = new ArrayList<UserObject>();

+ ArrayList<Integer> objArray = new ArrayList<Integer>();

ArrayList<String> primArray = new ArrayList<String>();

int arraySize = 0;

- public UserObject(String cls, int id, String objType) {

+ public UserObject(String cls, int id) {

this.cls = cls;

this.id = id;

- this.objType = objType;

+ //this.objType = objType;

+ }

+

+ public UserObject() {

+ // TODO Auto-generated constructor stub

}

}

commit d870ce8902610814137a9f2a6f65dd1b83a3006e

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 16:34:12

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 16:34:12

Parent: 3ba336932ea005e53ff37412a4be285930bb194a (Removed type code in UserObject)

Child: eed7424aa97a38599a5201502799d881439fd016 (Completed Visualizer)

Branches: master, origin/master

Completed Deserializer

diff --git a/src/Deserializer.java b/src/Deserializer.java

new file mode 100644

index 0000000..ae06dc7

--- /dev/null

+++ b/src/Deserializer.java

@@ -0,0 +1,60 @@

+import java.util.ArrayList;

+import java.util.List;

+

+import org.jdom2.\*;

+

+public class Deserializer {

+ Element root;

+ List<UserObject> objList = new ArrayList<UserObject>();

+

+

+ public Object deserialize(Document doc) {

+ root = doc.getRootElement();

+ List<Element> children = root.getChildren();

+ for (Element c : children) {

+ objList.add(deserialObj(c));

+ }

+

+ return new Object();

+ }

+

+ public UserObject deserialObj(Element ele) {

+

+ String cls = ele.getAttributeValue("class");

+ int id = Integer.parseInt(ele.getAttributeValue("id"));

+ String arrayLenS = ele.getAttributeValue("length");

+ int arrayLen = 0;

+ if (arrayLenS != null) {

+ arrayLen = Integer.parseInt(arrayLenS);

+ }

+ UserObject obj = new UserObject();

+ obj.cls = cls;

+ obj.id = id;

+ obj.arraySize = arrayLen;

+ List<Element> children = ele.getChildren();

+ for (Element c : children) {

+ if (c.getName().equals("field")) {

+ String text = c.getChild("value").getText();

+ if (text != null) {

+ UserField f = new UserField(c.getAttributeValue("name"), c.getAttributeValue("declaringclass"), text);

+ obj.fields.add(f);

+ } else {

+ UserField f = new UserField(c.getAttributeValue("name"), c.getAttributeValue("declaringclass"), Integer.parseInt(c.getChildText("reference")));

+ obj.fields.add(f);

+ }

+

+ }

+ else if (c.getName().equals("value")) {

+ obj.primArray.add(c.getText());

+ }

+ else {

+ obj.objArray.add(Integer.parseInt(c.getText()));

+ }

+ }

+

+

+

+

+ return obj;

+ }

+}

diff --git a/src/Receiver.java b/src/Receiver.java

index 7738f76..0ad7f38 100644

--- a/src/Receiver.java

+++ b/src/Receiver.java

@@ -1,10 +1,18 @@

import org.jdom2.\*;

+import org.xml.sax.SAXException;

+

import java.util.List;

+

+import javax.xml.parsers.DocumentBuilder;

+import javax.xml.parsers.DocumentBuilderFactory;

+import javax.xml.parsers.ParserConfigurationException;

+

import java.io.BufferedReader;

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.UnknownHostException;

+import java.nio.CharBuffer;

import java.io.InputStreamReader;

import java.io.PrintWriter;

@@ -12,19 +20,31 @@

private static ServerSocket serverSocket;

private static Socket clientSocket;

private static int port = 4444;

- private static PrintWriter out;

+ //private static PrintWriter out;

private static BufferedReader in;

- public void deserializer() {

-

- }

+ public static Document recieve() throws IOException, UnknownHostException, ParserConfigurationException, SAXException {

+ //send over socket

+ serverSocket = new ServerSocket(port);

+ clientSocket = serverSocket.accept();

+ //out = new PrintWriter(clientSocket.getOutputStream(), true);

+ in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

+ String recieved = "";

+ String readL = "";

+ while ((readL = in.readLine() + "\n") != null) {

+ recieved += readL;

+ }

+ DocumentBuilder db = DocumentBuilderFactory.newInstance().newDocumentBuilder();

+ Document doc = (Document)db.parse(recieved);

+ return doc;

+ }

- public static void main() throws IOException {

- //send over socket

- serverSocket = new ServerSocket(port);

- clientSocket = serverSocket.accept();

- out = new PrintWriter(clientSocket.getOutputStream(), true);

- in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

+ public static void main() throws IOException, UnknownHostException, ParserConfigurationException, SAXException{

+ Document doc = recieve();

+ Deserializer ds = new Deserializer();

+ ds.deserialize(doc);

+

+

}

diff --git a/src/Visualizer.java b/src/Visualizer.java

new file mode 100644

index 0000000..9004091

--- /dev/null

+++ b/src/Visualizer.java

@@ -0,0 +1,10 @@

+import java.util.List;

+

+public class Visualizer {

+

+ public void display(List<UserObject> objList) {

+ for (UserObject obj : objList) {

+

+ }

+ }

+}

commit eed7424aa97a38599a5201502799d881439fd016

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 16:55:50

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 16:55:50

Parent: d870ce8902610814137a9f2a6f65dd1b83a3006e (Completed Deserializer)

Child: 247fdb475e9ef4ebfed1ef807145fd524a57bf13 (Fixed a bug)

Branches: master, origin/master

Completed Visualizer

diff --git a/src/Deserializer.java b/src/Deserializer.java

index ae06dc7..71b5ca6 100644

--- a/src/Deserializer.java

+++ b/src/Deserializer.java

@@ -39,8 +39,11 @@

UserField f = new UserField(c.getAttributeValue("name"), c.getAttributeValue("declaringclass"), text);

obj.fields.add(f);

} else {

- UserField f = new UserField(c.getAttributeValue("name"), c.getAttributeValue("declaringclass"), Integer.parseInt(c.getChildText("reference")));

- obj.fields.add(f);

+ text = c.getChildText("reference");

+ if (!(text.contains("NULL"))) {

+ UserField f = new UserField(c.getAttributeValue("name"), c.getAttributeValue("declaringclass"), Integer.parseInt(text));

+ obj.fields.add(f);

+ }

}

}

diff --git a/src/Visualizer.java b/src/Visualizer.java

index 9004091..c2a26d3 100644

--- a/src/Visualizer.java

+++ b/src/Visualizer.java

@@ -3,8 +3,47 @@

public class Visualizer {

public void display(List<UserObject> objList) {

+ String str = "";

for (UserObject obj : objList) {

-

+ str += "Class: " + obj.cls + "\n";

+ str += "id#: " + obj.id + "\n";

+ if (obj.cls.contains("[")) {

+ str += "array length: " + obj.arraySize + "\n";

+ for (int i = 0; i < obj.arraySize; i++) {

+ if (i < obj.objArray.size()) {

+ str += ind(1) + "ObjectID: " + obj.objArray.get(i) + "\n";

+ }

+ else if (i < obj.primArray.size()) {

+ str += ind(1) + "value: " + obj.primArray.get(i) + "\n";

+ }

+ else {

+ str += ind(1) + "Null" + "\n";

+ }

+ }

+ }

+ else {

+ for (int i = 0; i < obj.objArray.size(); i++) {

+ str += ind(1) + "ObjectID: " + obj.objArray.get(i) + "\n";

+ }

+ for (UserField f : obj.fields) {

+ str += ind(1) + "Field " + f.name;

+ if (f.isPrim) {

+ str += " is " + f.value + "\n";

+ }

+ else {

+ str += " is " + f.reference + "\n";

+ }

+ }

+ }

}

+ System.out.println(str);

+ }

+

+ public String ind(int num) {

+ String ret = "";

+ for (int i = 0; i <= num; i++) {

+ ret += " ";

+ }

+ return ret;

}

}

commit 247fdb475e9ef4ebfed1ef807145fd524a57bf13

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 17:56:45

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 17:56:45

Parent: eed7424aa97a38599a5201502799d881439fd016 (Completed Visualizer)

Child: b1c50e9ddfa846394ee4796165289f7312a8f5a0 (Bug fixes)

Branches: master, origin/master

Fixed a bug

diff --git a/src/ObjectCreator.java b/src/ObjectCreator.java

index 21431f5..4adf6db 100644

--- a/src/ObjectCreator.java

+++ b/src/ObjectCreator.java

@@ -246,8 +246,8 @@

public void prepareNextID() {

nextID++;

- boolean tryAgain = false;

- while (!tryAgain) {

+ boolean tryAgain = true;

+ while (tryAgain) {

tryAgain = false;

for (UserObject obj : objList) {

if (obj.id == nextID) {

diff --git a/src/Receiver.java b/src/Receiver.java

index 0ad7f38..7a73f3b 100644

--- a/src/Receiver.java

+++ b/src/Receiver.java

@@ -31,15 +31,15 @@

in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

String recieved = "";

String readL = "";

- while ((readL = in.readLine() + "\n") != null) {

- recieved += readL;

+ while ((readL = in.readLine()) != null) {

+ recieved += readL + "\n";

}

DocumentBuilder db = DocumentBuilderFactory.newInstance().newDocumentBuilder();

Document doc = (Document)db.parse(recieved);

return doc;

}

- public static void main() throws IOException, UnknownHostException, ParserConfigurationException, SAXException{

+ public static void main(String[] args) throws IOException, UnknownHostException, ParserConfigurationException, SAXException{

Document doc = recieve();

Deserializer ds = new Deserializer();

ds.deserialize(doc);

commit b1c50e9ddfa846394ee4796165289f7312a8f5a0

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 20:47:39

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 20:47:39

Parent: 247fdb475e9ef4ebfed1ef807145fd524a57bf13 (Fixed a bug)

Child: dcabe96f20cea59df2f41319206f2810d73972cf (Test cases for Serializer Completed)

Branches: master, origin/master

Bug fixes

diff --git a/src/Deserializer.java b/src/Deserializer.java

index 71b5ca6..e1b03c3 100644

--- a/src/Deserializer.java

+++ b/src/Deserializer.java

@@ -34,23 +34,23 @@

List<Element> children = ele.getChildren();

for (Element c : children) {

if (c.getName().equals("field")) {

- String text = c.getChild("value").getText();

+ String text = c.getChildText("value");

if (text != null) {

- UserField f = new UserField(c.getAttributeValue("name"), c.getAttributeValue("declaringclass"), text);

+ UserField f = new UserField(c.getAttributeValue("declaringclass"), c.getAttributeValue("name"), text);

obj.fields.add(f);

} else {

text = c.getChildText("reference");

if (!(text.contains("NULL"))) {

- UserField f = new UserField(c.getAttributeValue("name"), c.getAttributeValue("declaringclass"), Integer.parseInt(text));

+ UserField f = new UserField(c.getAttributeValue("declaringclass"), c.getAttributeValue("name"), Integer.parseInt(text));

obj.fields.add(f);

}

}

}

- else if (c.getName().equals("value")) {

+ else if ((obj.primArray.size() < obj.arraySize) && (c.getName().equals("value"))) {

obj.primArray.add(c.getText());

}

- else {

+ else if ((obj.objArray.size() < obj.arraySize) && !(c.getText().contains("NULL"))){

obj.objArray.add(Integer.parseInt(c.getText()));

}

}

diff --git a/src/Receiver.java b/src/Receiver.java

index 7a73f3b..5f0ab6c 100644

--- a/src/Receiver.java

+++ b/src/Receiver.java

@@ -1,5 +1,8 @@

import org.jdom2.\*;

+import org.jdom2.input.SAXBuilder;

import org.xml.sax.SAXException;

+

+import jdk.internal.org.xml.sax.InputSource;

import java.util.List;

@@ -15,6 +18,7 @@

import java.nio.CharBuffer;

import java.io.InputStreamReader;

import java.io.PrintWriter;

+import java.io.StringReader;

public class Receiver {

private static ServerSocket serverSocket;

@@ -23,28 +27,41 @@

//private static PrintWriter out;

private static BufferedReader in;

+ public static void setup() throws IOException {

+ serverSocket = new ServerSocket(port);

+ }

+

public static Document recieve() throws IOException, UnknownHostException, ParserConfigurationException, SAXException {

//send over socket

- serverSocket = new ServerSocket(port);

+

clientSocket = serverSocket.accept();

//out = new PrintWriter(clientSocket.getOutputStream(), true);

in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

String recieved = "";

String readL = "";

while ((readL = in.readLine()) != null) {

- recieved += readL + "\n";

+ recieved += readL;

}

- DocumentBuilder db = DocumentBuilderFactory.newInstance().newDocumentBuilder();

- Document doc = (Document)db.parse(recieved);

+ org.jdom2.input.SAXBuilder saxBuilder = new SAXBuilder();

+ org.jdom2.Document doc = null;

+ try {

+ doc = saxBuilder.build(new StringReader(recieved));

+ } catch (Exception e) {

+ System.out.println(e);

+ System.exit(-1);

+ }

return doc;

}

public static void main(String[] args) throws IOException, UnknownHostException, ParserConfigurationException, SAXException{

- Document doc = recieve();

- Deserializer ds = new Deserializer();

- ds.deserialize(doc);

-

-

+ setup();

+ while (true) {

+ Document doc = recieve();

+ Deserializer ds = new Deserializer();

+ ds.deserialize(doc);

+ Visualizer v = new Visualizer();

+ v.display(ds.objList);

+ }

}

diff --git a/src/Sender.java b/src/Sender.java

index 29d89d0..53ef500 100644

--- a/src/Sender.java

+++ b/src/Sender.java

@@ -12,12 +12,12 @@

public class Sender {

private static Socket clientSocket;

private static PrintWriter out;

- private static BufferedReader in;

+ //private static BufferedReader in;

public static void startConnection(String ip, int port) throws UnknownHostException, IOException {

- clientSocket = new Socket(ip, port);

+ clientSocket = new Socket(ip, port);

out = new PrintWriter(clientSocket.getOutputStream(), true);

- in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

+ //in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

}

/\*public String sendMessage(String msg) throws IOException {

@@ -27,7 +27,7 @@

}\*/

public static void stopConnection() throws IOException {

- in.close();

+ //in.close();

out.close();

clientSocket.close();

}

@@ -39,9 +39,9 @@

Serializer s = new Serializer();

org.jdom2.Document doc = s.serialize(objList);

- startConnection("127.0.0.1", 4444);

+ startConnection(args[0], 4444);

out.println(new XMLOutputter().outputString(doc));

-

-

+ stopConnection();

+ System.out.println(new XMLOutputter().outputString(doc));

}

}

diff --git a/src/Serializer.java b/src/Serializer.java

index eda764f..36255ff 100644

--- a/src/Serializer.java

+++ b/src/Serializer.java

@@ -45,25 +45,26 @@

ele.addContent(e);

}

}

- else if (obj.primArray.size() > 0 && obj.arraySize > 0) {

+ else if (obj.arraySize > 0) {

ele.setAttribute("length", String.valueOf(obj.arraySize));

- for (int i = 0; i < obj.arraySize; i++) {

- if (i < obj.primArray.size()) {

- ele.addContent((new Element("value")).setText(obj.primArray.get(i)));

- }

- else {

- ele.addContent((new Element("value")).setText("0"));

+ if (obj.cls.contains("Object")) {

+ for (int i = 0; i < obj.arraySize; i++) {

+ if (i < obj.objArray.size()) {

+ ele.addContent((new Element("reference")).setText(String.valueOf(obj.objArray.get(i))));

+ }

+ else {

+ ele.addContent((new Element("reference")).setText("NULL"));

+ }

}

}

- }

- else if (obj.objArray.size() > 0 && obj.arraySize > 0) {

- ele.setAttribute("length", String.valueOf(obj.arraySize));

- for (int i = 0; i < obj.arraySize; i++) {

- if (i < obj.objArray.size()) {

- ele.addContent((new Element("reference")).setText(String.valueOf(obj.objArray.get(i))));

- }

- else {

- ele.addContent((new Element("reference")).setText("NULL"));

+ else {

+ for (int i = 0; i < obj.arraySize; i++) {

+ if (i < obj.primArray.size()) {

+ ele.addContent((new Element("value")).setText(obj.primArray.get(i)));

+ }

+ else {

+ ele.addContent((new Element("value")).setText("0"));

+ }

}

}

}

diff --git a/src/Visualizer.java b/src/Visualizer.java

index c2a26d3..a106629 100644

--- a/src/Visualizer.java

+++ b/src/Visualizer.java

@@ -28,15 +28,17 @@

for (UserField f : obj.fields) {

str += ind(1) + "Field " + f.name;

if (f.isPrim) {

- str += " is " + f.value + "\n";

+ str += " has a value of " + f.value + "\n";

}

else {

- str += " is " + f.reference + "\n";

+ str += " is id#" + f.reference + "\n";

}

}

}

+ str += "\n";

}

System.out.println(str);

+ System.out.println("------------------------------------\n\n");

}

public String ind(int num) {

commit dcabe96f20cea59df2f41319206f2810d73972cf

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 22:10:10

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 22:10:10

Parent: b1c50e9ddfa846394ee4796165289f7312a8f5a0 (Bug fixes)

Child: 6d00b110403b4786b37ac1efbb6bda54764444f5 (Completed Deserializer tests. Fixed issue in collection recreation)

Branches: master, origin/master

Test cases for Serializer Completed

diff --git a/src/Serializer.java b/src/Serializer.java

index 36255ff..0359e50 100644

--- a/src/Serializer.java

+++ b/src/Serializer.java

@@ -9,7 +9,7 @@

public org.jdom2.Document serialize(Object norm\_obj) {

if (!(norm\_obj instanceof List<?>)) {

- System.out.println("cannot serialize an object that is not a UserObject");

+ System.out.println("cannot serialize an object that is not a List of UserObject");

System.exit(-1);

}

@SuppressWarnings("unchecked")

diff --git a/src/TesterSerializerDeserializer.java b/src/TesterSerializerDeserializer.java

new file mode 100644

index 0000000..d67b20d

--- /dev/null

+++ b/src/TesterSerializerDeserializer.java

@@ -0,0 +1,126 @@

+import org.junit.Assert.\*;

+import org.junit.Before;

+

+import static org.junit.Assert.\*;

+

+import java.util.ArrayList;

+import java.util.List;

+

+import org.jdom2.Document;

+import org.jdom2.output.XMLOutputter;

+import org.junit.After;

+import org.junit.Test;

+

+public class TesterSerializerDeserializer {

+

+ Serializer s;

+ Deserializer ds;

+ Visualizer v;

+ UserObject oP;

+ UserObject oR;

+ UserObject aP;

+ UserObject aO;

+ UserObject col;

+

+ @Before

+ public void before() {

+ s = new Serializer();

+ ds = new Deserializer();

+ v = new Visualizer();

+ oP = new UserObject("oP", 0);

+ for (int i = 1; i < 2; i++) {

+ oP.fields.add(new UserField(oP.cls, "f" + i, String.valueOf(i)));

+ }

+ oR = new UserObject("oR", 1);

+ for (int i = 1; i < 2; i++) {

+ oR.fields.add(new UserField(oR.cls, "f" + i, oR.id));

+ }

+ aP = new UserObject("[F", 2);

+ aP.arraySize = 3;

+ for (int i = 0; i < 2; i++) {

+ aP.primArray.add(String.valueOf(i));

+ }

+ aO = new UserObject("[Object", 3);

+ aO.arraySize = 3;

+ for (int i = 0; i < 2; i++) {

+ aO.objArray.add(3);

+ }

+ col = new UserObject(ArrayList.class.getName(), 4);

+ for (int i = 0; i < 2; i++) {

+ col.objArray.add(4);

+ }

+

+ }

+

+ @After

+ public void after() {

+ s = null;

+ ds = null;

+ v = null;

+ oP = null;

+ oR = null;

+ aP = null;

+ aO = null;

+ col = null;

+ }

+

+ @Test

+ public void testSerializeOP() {

+ List<UserObject> list = new ArrayList<UserObject>();

+ list.add(oP);

+ Document doc = s.serialize(list);

+ String str = new XMLOutputter().outputString(doc);

+ System.out.println(str);

+ assertTrue(str.contains(serOP));

+ }

+

+

+ @Test

+ public void testSerializeOR() {

+ List<UserObject> list = new ArrayList<UserObject>();

+ list.add(oR);

+ Document doc = s.serialize(list);

+ String str = new XMLOutputter().outputString(doc);

+ System.out.println(str);

+ assertTrue(str.contains(serOR));

+ }

+

+ @Test

+ public void testSerializeAP() {

+ List<UserObject> list = new ArrayList<UserObject>();

+ list.add(aP);

+ Document doc = s.serialize(list);

+ String str = new XMLOutputter().outputString(doc);

+ System.out.println(str);

+ assertTrue(str.contains(serAP));

+ }

+

+ @Test

+ public void testSerializeAO() {

+ List<UserObject> list = new ArrayList<UserObject>();

+ list.add(aO);

+ Document doc = s.serialize(list);

+ String str = new XMLOutputter().outputString(doc);

+ System.out.println(str);

+ assertTrue(str.contains(serAO));

+ }

+

+ @Test

+ public void testSerializeCol() {

+ List<UserObject> list = new ArrayList<UserObject>();

+ list.add(col);

+ Document doc = s.serialize(list);

+ String str = new XMLOutputter().outputString(doc);

+ System.out.println(str);

+ assertTrue(str.contains(serCol));

+ }

+

+

+

+ String serOP = "<serialized><object class=\"oP\" id=\"0\"><field name=\"f1\" declaringclass=\"oP\"><value>1</value></field></object></serialized>";

+ String serOR = "<serialized><object class=\"oR\" id=\"1\"><field name=\"f1\" declaringclass=\"oR\"><reference>1</reference></field></object></serialized>";

+ String serAP = "<serialized><object class=\"[F\" id=\"2\" length=\"3\"><value>0</value><value>1</value><value>0</value></object></serialized>";

+ String serAO = "<serialized><object class=\"[Object\" id=\"3\" length=\"3\"><reference>3</reference><reference>3</reference><reference>NULL</reference></object></serialized>";

+ String serCol = "<serialized><object class=\"java.util.ArrayList\" id=\"4\"><reference>4</reference><reference>4</reference></object></serialized>";

+

+}

commit 6d00b110403b4786b37ac1efbb6bda54764444f5

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 22:38:49

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 22:38:49

Parent: dcabe96f20cea59df2f41319206f2810d73972cf (Test cases for Serializer Completed)

Child: a138c0bda8c5759d6090597d5bf882e9db13bcc1 (Refactored serial(). Removed Long method bad smell)

Branches: master, origin/master

Completed Deserializer tests. Fixed issue in collection recreation

diff --git a/src/Deserializer.java b/src/Deserializer.java

index e1b03c3..3ee3f8a 100644

--- a/src/Deserializer.java

+++ b/src/Deserializer.java

@@ -15,7 +15,7 @@

objList.add(deserialObj(c));

}

- return new Object();

+ return objList;

}

public UserObject deserialObj(Element ele) {

@@ -50,7 +50,7 @@

else if ((obj.primArray.size() < obj.arraySize) && (c.getName().equals("value"))) {

obj.primArray.add(c.getText());

}

- else if ((obj.objArray.size() < obj.arraySize) && !(c.getText().contains("NULL"))){

+ else if (!(c.getText().contains("NULL"))){

obj.objArray.add(Integer.parseInt(c.getText()));

}

}

diff --git a/src/TesterSerializerDeserializer.java b/src/TesterSerializerDeserializer.java

index d67b20d..bc28436 100644

--- a/src/TesterSerializerDeserializer.java

+++ b/src/TesterSerializerDeserializer.java

@@ -27,24 +27,30 @@

s = new Serializer();

ds = new Deserializer();

v = new Visualizer();

+

oP = new UserObject("oP", 0);

for (int i = 1; i < 2; i++) {

oP.fields.add(new UserField(oP.cls, "f" + i, String.valueOf(i)));

}

+

oR = new UserObject("oR", 1);

for (int i = 1; i < 2; i++) {

oR.fields.add(new UserField(oR.cls, "f" + i, oR.id));

}

+

aP = new UserObject("[F", 2);

aP.arraySize = 3;

for (int i = 0; i < 2; i++) {

aP.primArray.add(String.valueOf(i));

}

+ aP.primArray.add(String.valueOf(0));

+

aO = new UserObject("[Object", 3);

aO.arraySize = 3;

for (int i = 0; i < 2; i++) {

aO.objArray.add(3);

}

+

col = new UserObject(ArrayList.class.getName(), 4);

for (int i = 0; i < 2; i++) {

col.objArray.add(4);

@@ -70,7 +76,7 @@

list.add(oP);

Document doc = s.serialize(list);

String str = new XMLOutputter().outputString(doc);

- System.out.println(str);

+ //System.out.println(str);

assertTrue(str.contains(serOP));

}

@@ -81,7 +87,7 @@

list.add(oR);

Document doc = s.serialize(list);

String str = new XMLOutputter().outputString(doc);

- System.out.println(str);

+ //System.out.println(str);

assertTrue(str.contains(serOR));

}

@@ -91,7 +97,7 @@

list.add(aP);

Document doc = s.serialize(list);

String str = new XMLOutputter().outputString(doc);

- System.out.println(str);

+ //System.out.println(str);

assertTrue(str.contains(serAP));

}

@@ -101,7 +107,7 @@

list.add(aO);

Document doc = s.serialize(list);

String str = new XMLOutputter().outputString(doc);

- System.out.println(str);

+ //System.out.println(str);

assertTrue(str.contains(serAO));

}

@@ -111,10 +117,162 @@

list.add(col);

Document doc = s.serialize(list);

String str = new XMLOutputter().outputString(doc);

- System.out.println(str);

+ //System.out.println(str);

assertTrue(str.contains(serCol));

}

+ @Test

+ public void testDeserializeOP() {

+ List<UserObject> list = new ArrayList<UserObject>();

+ list.add(oP);

+ Document doc = s.serialize(list);

+ UserObject obj = ((List<UserObject>) ds.deserialize(doc)).get(0);

+ UserObject orig = oP;

+ assertEquals(orig.id, obj.id);

+ assertEquals(orig.cls, obj.cls);

+ assertEquals(orig.arraySize, obj.arraySize);

+ assertEquals(orig.fields.size(), obj.fields.size());

+ assertEquals(orig.primArray.size(), obj.primArray.size());

+ assertEquals(orig.objArray.size(), obj.objArray.size());

+ for (int i = 0; i < orig.fields.size(); i++) {

+ UserField f = orig.fields.get(i);

+ assertEquals(f.declaringclass, obj.fields.get(i).declaringclass);

+ assertEquals(f.name, obj.fields.get(i).name);

+ assertEquals(f.isObject, obj.fields.get(i).isObject);

+ assertEquals(f.isPrim, obj.fields.get(i).isPrim);

+ assertEquals(f.reference, obj.fields.get(i).reference);

+ assertEquals(f.value, obj.fields.get(i).value);

+ }

+ for (int i = 0; i < orig.primArray.size(); i++) {

+ assertEquals(orig.primArray.get(i), obj.primArray.get(i));

+ }

+ for (int i = 0; i < orig.objArray.size(); i++) {

+ assertEquals(orig.objArray.get(i), obj.objArray.get(i));

+ }

+

+ }

+

+

+ @Test

+ public void testDeserializeOR() {

+ List<UserObject> list = new ArrayList<UserObject>();

+ list.add(oR);

+ Document doc = s.serialize(list);

+ UserObject obj = ((List<UserObject>) ds.deserialize(doc)).get(0);

+ UserObject orig = oR;

+ assertEquals(orig.id, obj.id);

+ assertEquals(orig.cls, obj.cls);

+ assertEquals(orig.arraySize, obj.arraySize);

+ assertEquals(orig.fields.size(), obj.fields.size());

+ assertEquals(orig.primArray.size(), obj.primArray.size());

+ assertEquals(orig.objArray.size(), obj.objArray.size());

+ for (int i = 0; i < orig.fields.size(); i++) {

+ UserField f = orig.fields.get(i);

+ assertEquals(f.declaringclass, obj.fields.get(i).declaringclass);

+ assertEquals(f.name, obj.fields.get(i).name);

+ assertEquals(f.isObject, obj.fields.get(i).isObject);

+ assertEquals(f.isPrim, obj.fields.get(i).isPrim);

+ assertEquals(f.reference, obj.fields.get(i).reference);

+ assertEquals(f.value, obj.fields.get(i).value);

+ }

+ for (int i = 0; i < orig.primArray.size(); i++) {

+ assertEquals(orig.primArray.get(i), obj.primArray.get(i));

+ }

+ for (int i = 0; i < orig.objArray.size(); i++) {

+ assertEquals(orig.objArray.get(i), obj.objArray.get(i));

+ }

+ }

+

+ @Test

+ public void testDeserializeAP() {

+ List<UserObject> list = new ArrayList<UserObject>();

+ list.add(aP);

+ Document doc = s.serialize(list);

+ UserObject obj = ((List<UserObject>) ds.deserialize(doc)).get(0);

+ UserObject orig = aP;

+ assertEquals(orig.id, obj.id);

+ assertEquals(orig.cls, obj.cls);

+ assertEquals(orig.arraySize, obj.arraySize);

+ assertEquals(orig.fields.size(), obj.fields.size());

+ assertEquals(orig.primArray.size(), obj.primArray.size());

+ assertEquals(orig.objArray.size(), obj.objArray.size());

+ for (int i = 0; i < orig.fields.size(); i++) {

+ UserField f = orig.fields.get(i);

+ assertEquals(f.declaringclass, obj.fields.get(i).declaringclass);

+ assertEquals(f.name, obj.fields.get(i).name);

+ assertEquals(f.isObject, obj.fields.get(i).isObject);

+ assertEquals(f.isPrim, obj.fields.get(i).isPrim);

+ assertEquals(f.reference, obj.fields.get(i).reference);

+ assertEquals(f.value, obj.fields.get(i).value);

+ }

+ for (int i = 0; i < orig.primArray.size(); i++) {

+ assertEquals(orig.primArray.get(i), obj.primArray.get(i));

+ }

+ for (int i = 0; i < orig.objArray.size(); i++) {

+ assertEquals(orig.objArray.get(i), obj.objArray.get(i));

+ }

+ }

+

+ @Test

+ public void testDeserializeAO() {

+ List<UserObject> list = new ArrayList<UserObject>();

+ list.add(aO);

+ Document doc = s.serialize(list);

+ UserObject obj = ((List<UserObject>) ds.deserialize(doc)).get(0);

+ UserObject orig = aO;

+ assertEquals(orig.id, obj.id);

+ assertEquals(orig.cls, obj.cls);

+ assertEquals(orig.arraySize, obj.arraySize);

+ assertEquals(orig.fields.size(), obj.fields.size());

+ assertEquals(orig.primArray.size(), obj.primArray.size());

+ assertEquals(orig.objArray.size(), obj.objArray.size());

+ for (int i = 0; i < orig.fields.size(); i++) {

+ UserField f = orig.fields.get(i);

+ assertEquals(f.declaringclass, obj.fields.get(i).declaringclass);

+ assertEquals(f.name, obj.fields.get(i).name);

+ assertEquals(f.isObject, obj.fields.get(i).isObject);

+ assertEquals(f.isPrim, obj.fields.get(i).isPrim);

+ assertEquals(f.reference, obj.fields.get(i).reference);

+ assertEquals(f.value, obj.fields.get(i).value);

+ }

+ for (int i = 0; i < orig.primArray.size(); i++) {

+ assertEquals(orig.primArray.get(i), obj.primArray.get(i));

+ }

+ for (int i = 0; i < orig.objArray.size(); i++) {

+ assertEquals(orig.objArray.get(i), obj.objArray.get(i));

+ }

+ }

+

+ @Test

+ public void testDeserializeCol() {

+ List<UserObject> list = new ArrayList<UserObject>();

+ list.add(col);

+ Document doc = s.serialize(list);

+ UserObject obj = ((List<UserObject>) ds.deserialize(doc)).get(0);

+ UserObject orig = col;

+ assertEquals(orig.id, obj.id);

+ assertEquals(orig.cls, obj.cls);

+ assertEquals(orig.arraySize, obj.arraySize);

+ assertEquals(orig.fields.size(), obj.fields.size());

+ assertEquals(orig.primArray.size(), obj.primArray.size());

+ assertEquals(orig.objArray.size(), obj.objArray.size());

+ for (int i = 0; i < orig.fields.size(); i++) {

+ UserField f = orig.fields.get(i);

+ assertEquals(f.declaringclass, obj.fields.get(i).declaringclass);

+ assertEquals(f.name, obj.fields.get(i).name);

+ assertEquals(f.isObject, obj.fields.get(i).isObject);

+ assertEquals(f.isPrim, obj.fields.get(i).isPrim);

+ assertEquals(f.reference, obj.fields.get(i).reference);

+ assertEquals(f.value, obj.fields.get(i).value);

+ }

+ for (int i = 0; i < orig.primArray.size(); i++) {

+ assertEquals(orig.primArray.get(i), obj.primArray.get(i));

+ }

+ for (int i = 0; i < orig.objArray.size(); i++) {

+ assertEquals(orig.objArray.get(i), obj.objArray.get(i));

+ }

+ }

+

String serOP = "<serialized><object class=\"oP\" id=\"0\"><field name=\"f1\" declaringclass=\"oP\"><value>1</value></field></object></serialized>";

commit a138c0bda8c5759d6090597d5bf882e9db13bcc1

Author: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 22:45:19

Committer: ttdjohns <32712899+ttdjohns@users.noreply.github.com> 2019-03-21 22:45:19

Parent: 6d00b110403b4786b37ac1efbb6bda54764444f5 (Completed Deserializer tests. Fixed issue in collection recreation)

Branches: master, origin/master

Refactored serial(). Removed Long method bad smell

diff --git a/src/Serializer.java b/src/Serializer.java

index 0359e50..ea2ede1 100644

--- a/src/Serializer.java

+++ b/src/Serializer.java

@@ -28,45 +28,13 @@

done.add(obj.id);

if (obj.fields.size() > 0 && obj.fields.get(0).isPrim) {

- for (UserField i : obj.fields) {

- Element e = new Element("field");

- e.setAttribute("name", i.name);

- e.setAttribute("declaringclass", i.declaringclass);

- e.addContent((new Element("value")).setText(i.value));

- ele.addContent(e);

- }

+ serialPrimFields(obj, ele);

}

else if (obj.fields.size() > 0 && obj.fields.get(0).isObject) {

- for (UserField i : obj.fields) {

- Element e = new Element("field");

- e.setAttribute("name", i.name);

- e.setAttribute("declaringclass", i.declaringclass);

- e.addContent((new Element("reference")).setText(String.valueOf(i.reference)));

- ele.addContent(e);

- }

+ serialObjFields(obj, ele);

}

else if (obj.arraySize > 0) {

- ele.setAttribute("length", String.valueOf(obj.arraySize));

- if (obj.cls.contains("Object")) {

- for (int i = 0; i < obj.arraySize; i++) {

- if (i < obj.objArray.size()) {

- ele.addContent((new Element("reference")).setText(String.valueOf(obj.objArray.get(i))));

- }

- else {

- ele.addContent((new Element("reference")).setText("NULL"));

- }

- }

- }

- else {

- for (int i = 0; i < obj.arraySize; i++) {

- if (i < obj.primArray.size()) {

- ele.addContent((new Element("value")).setText(obj.primArray.get(i)));

- }

- else {

- ele.addContent((new Element("value")).setText("0"));

- }

- }

- }

+ serialArrays(obj, ele);

}

else {

for (int i = 0; i < obj.objArray.size(); i++) {

@@ -76,4 +44,51 @@

root.addContent(ele);

}

+

+ private void serialArrays(UserObject obj, Element ele) {

+ ele.setAttribute("length", String.valueOf(obj.arraySize));

+ if (obj.cls.contains("Object")) {

+ for (int i = 0; i < obj.arraySize; i++) {

+ if (i < obj.objArray.size()) {

+ ele.addContent((new Element("reference")).setText(String.valueOf(obj.objArray.get(i))));

+ }

+ else {

+ ele.addContent((new Element("reference")).setText("NULL"));

+ }

+ }

+ }

+ else {

+ for (int i = 0; i < obj.arraySize; i++) {

+ if (i < obj.primArray.size()) {

+ ele.addContent((new Element("value")).setText(obj.primArray.get(i)));

+ }

+ else {

+ ele.addContent((new Element("value")).setText("0"));

+ }

+ }

+ }

+

+ }

+

+ private void serialObjFields(UserObject obj, Element ele) {

+ for (UserField i : obj.fields) {

+ Element e = new Element("field");

+ e.setAttribute("name", i.name);

+ e.setAttribute("declaringclass", i.declaringclass);

+ e.addContent((new Element("reference")).setText(String.valueOf(i.reference)));

+ ele.addContent(e);

+ }

+

+ }

+

+ private void serialPrimFields(UserObject obj, Element ele) {

+ for (UserField i : obj.fields) {

+ Element e = new Element("field");

+ e.setAttribute("name", i.name);

+ e.setAttribute("declaringclass", i.declaringclass);

+ e.addContent((new Element("value")).setText(i.value));

+ ele.addContent(e);

+ }

+

+ }

}