CS146Section8Lab1 Name: Michael Huang Due date: February 8th, 2016

Ex-1: What are the values of the following?

1. 2 + 2 + "2"

Code:

public class Problem01A {

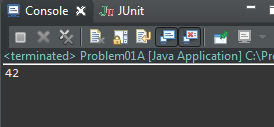
public static void main(String args[]) {

System.out.println(2 + 2 + "2");

}

}

Console output:



The 42 is not a String of “42” or the number 42. The 4 is a result of printing out 2 + 2, which results in 4, and printing a String “2” after it. This will result in an integer 4 and a String of “2”, hence 42.

1. "" + Countries, where countries is an ArrayList ﬁlled with several strings.

Code:

import java.util.ArrayList;

public class Problem01B {

public static void main(String args[]) {

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

countries.add("USA");

countries.add("China");

countries.add("Russia");

countries.add("France");

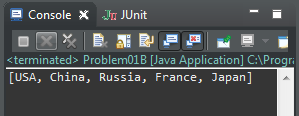
countries.add("Japan");

System.out.println("" + countries);

}

}

Console output:



Printing “” yields nothing. Printing the ArrayList country will invoke its toString method, which will yield its elements in brackets.

1. Hello" + new Greeter ("World").

Code:

/\*\*

\* A class for producing simple greetings.

\*/

public class Greeter {

private String name;

/\*\*

\* Constructs a Greeter object that can greet a person or entity.

\*/

public Greeter(String aName) {

name = aName;

}

/\*\*

\* Greet with a "Hello" message.

\* @return a message containing "Hello" and the name of the person

\* or entity.

\*/

public String sayHello() {

return "Hello, " + name + "!";

}

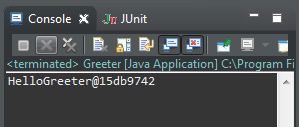
public static void main(String[] args) {

System.out.println("Hello" + new Greeter("World"));

}

}

Console output:



Printing “Hello” yields the String “Hello”. However, printing the greeter will merely yield its hashcode because there is no toString method in the Greeter class overriding Object’s toString method.

Ex-2: Write a simple generic version of method isEqualTo that compares its two arguments with the equals method and returns true if they’re equal and false otherwise. Use this generic method in a program that calls isEqualTo with a variety of built-in types, such as Object or Integer. What result do you get when you attempt to run this program?

Code:

public class Problem02 {

public static <T> boolean isEqualTo(T obj1, T obj2) {

return (obj1.equals(obj2));

}

public static void main(String[] args) {

int a = 1;

Integer b = new Integer(1);

Object c = new Object();

int d = a;

System.out.println(isEqualTo(a, b));

System.out.println(isEqualTo(c, d));

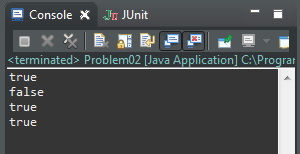
System.out.println(isEqualTo(a, d));

System.out.println(isEqualTo(d, b));

}

}

Console output:



a is equal to b due to unboxing. c and d are not equal because one is an object, the other is a primitive int. a and d are equal because they have the same reference thus have the same values. b and d are equal due to unboxing and d being the same as a, is the same.

Ex-3: Write a generic class Pair which has two type parameters—F and S— each representing the type of the first and second element of the pair, respectively. Add get and set methods for the first and second elements of the pair.

[Hint: The class header should be public class Pair <F, S>.]

Code:

public class Pair <F, S> {

private F first;

private S second;

public Pair(F f, S s) {

first = f;

second = s;

}

public void setFirst(F f) {

first = f;

}

public void setSecond(S s) {

second = s;

}

public F getFirst() {

return first;

}

public S getSecond() {

return second;

}

public static void main(String[] args) {

Pair<Integer, String> pair =

new Pair<Integer, String>(420, "Blaze it");

System.out.println(pair.getFirst() + " " +

pair.getSecond());

pair.setFirst(1048596);

pair.setSecond("Open the Steins Gate");

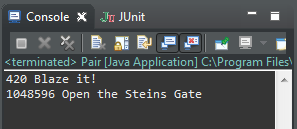
System.out.println(pair.getFirst() + " " +

pair.getSecond());

}

}

Console output:



The main method tests the set and get methods for the first and second elements.