Prepared by Ethel Tshukudu
Project: Programming Language Transfer
Centre for Computing Science Education/ University of Glasgow

Seminar 1: IN1010
University of Oslo
Java Pre Quiz answers

Question 1

Java	Answer	Python	Answer	Does the Java code snippet mean the same as the Python code? Why?

`int a = 3;`
`String b = "fi";`
`double c = 10.2;`
`int[] e = {1, 2, 3};`
`int[] f = {1, 2, 3};`
		`a = 3`
		`b = "fi";`
		`c = 10.2;`
		`e = [1, 2, 3]`
		`f = [1, 2, 3]`		

`System.out.println(e==f);`	**`False`**	`print(e==f)`	True	Meaning is different *Java compares references (addresses) with ==. Python redefines (overloads) the == operator to compare **the objects** if __eq__ is defined for the class*
`System.out.println(e[0]+e[2]);`	**`4`**	`print(e[0]+e[2])`	4	Meaning is the same *Both are adding array values*
`System.out.println(a + b);`	**`3fi`**	`print(a + b)`	Error	Meaning is different *You can't **add** a **string** and an **integer**. But you can turn a number into a **string** if you use the **str**() in Python. Java does that conversion implicitly when '+' is used between a String and something of another type (it calls the appropriate class's **toString** method).*
`System.out.println(b + b);`	**`fifi`**	`print(b + b)`	fifi	Meaning is the same *They both allow string concatenation*

`System.out.println(a/2);`	**1**	`print(a/2)`	1.5	Meaning is different *Java handles this as integer division because both operands are integers. If any of the operands are double/ float normal division is performed in Java. Python always performs a normal division with the **/** operator, and treats the result as an integer or float value depending on result.*
`int sum = a + c;` `System.out.println(sum);`	**error**	`sum = a + c` `print(sum)`	13.2	Meaning is different *Java: once declared, type of a variable cannot be changed (static typing) vs dynamic typing in Python*
`System.out.println(a*(a+2));`	**15**	`print(a*(a+2))`	15	Meaning is the same *Precedence in these expressions works the same way*

Question 2

Java	Answer	Python	Answer	Does the Java code snippet mean the same as the Python code? Why?

`for (int i=0; i<2; i++){`
`  System.out.println("ola");`
`  i=3;`
`}`
	ola	`for i in range(2):`
		`  print ("ola")`
		`  i=3`
			ola
			ola
				Meaning is different
				*Index-based versus for-each (range) loop. NB: A **for-each loop** in Java would also "forget" any assignment made to **i** within the loop, when it starts a new iteration – then i gets the next value in the list/ array/ regardless of current value.*

`for (int i=0; i<2; i++){`
`  int a=2;`
`}`
`System.out.println(a);`
	error	`for i in range(2):`
		`  a=2`
		`print (a)`
			2	Meaning is different
				Scope: A variable declared inside pair of brackets "{" and "}" in a for-loop (and elsewhere) has scope within the brackets only.

Question 3

Prepared by Ethel Tshukudu
Project: Programming Language Transfer
Centre for Computing Science Education/ University of Glasgow

Java	Answer	Python	Answer	Does the Java code snippet mean the same as the Python code? Why?

`int sum = 0;`
`int i = 0;`
`while (i < 3){`
`  sum = sum + i;`
`  i++;`
`}`
`System.out.println(sum);`
	3	`total=0`
		`i = 0`
		`while (i < 3):`
		`  total +=i`
		`  i+=1`
		`print(total)`
			3	Meaning is the same
				The while loop in both languages executes a sequence of statements repeatedly while the condition is true(i<3)

Prepared by Ethel Tshukudu
Project: Programming Language Transfer
Centre for Computing Science Education/ University of Glasgow

Question 4

Java	Answer	Python	Answer	Does the Java code snippet mean the same as the Python code? Why?

`public class HelloWorld`
`{`
`  public static void main(String []args){`
`  System.out.println`
`  (map (7,"hello"));`
`  }`
 	
`  public static int map(int g, String s)`
`  {`
`  int a=g+2;`
`  can();`
`  return a;`
`  }`
 	
`  public static void can()`
`  {`
`  int a=3;`
`  System.out.println(a);`
`  }`
`}`
	3
	9
		`def cap(z,y):`
		`  pa=z+2`
		`  zap()`
		`  return pa`
		
		`def zap():`
		`  pa=3`
		`  print(pa)`
		
		`print(cap(7,"hello"))`
			3
			9
				Meaning is the same
				Methods with return
				
				*e.g (Java **map**), (Python **cap**)*
				Methods without return
				*e.g (Java **can**), (Python **zap**)*
				
				Methods calling and parameters and arguments
				*e.g. (Java **map(int g, String s), cap(7,"hello"))***
				*(Python **map(cap(z,y)), map (7,"hello"))***

Prepared by Ethel Tshukudu
Project: Programming Language Transfer
Centre for Computing Science Education/ University of Glasgow

Question 5

Java	Answer	Python	Answer	Does the Java code snippet mean the same as the Python code? Why?

`public class toya {`
`  int sweets;`
`  String name;`
 	
`  public toya (String nam, int sweets1){`
`  name=nam;`
`  sweets=sweets1;`
`  }`
`  public static void main(String args[]){`
`  toya f1=new toya("Fiki", 15);`
`  toya f2=new toya("Thabo", 15);`
`  f1.maya(10);`
`  f2.roko(5);`
`  System.out.println(f1.sweets + " " + f2.sweets);`
`  }`
`  public int maya(int a) {`
`  sweets=sweets-a;`
`  return sweets;`
`  }`
`  public int roko(int b) {`
`  sweets=sweets+b;`
`  return sweets;`
`  }}`
	5 20	`class Toya:`
		` def __init__(self, nam, sweets1):`
		`  self._name = nam`
		`  self._sweets = sweets1`
		
		` def maya(self, a):`
		`  self._sweets =self._sweets-a`
		`  return self._sweets`
		
		` def roko(self, b):`
		`  self._sweets =self._sweets+b`
		`  return self._sweets`
		
		`f1= Toya("Fiki", 15)`
		`f2= Toya("Thabo", 15)`
		`f1.maya(10)`
		`f2.roko(5)`
		`print (str(f1._sweets) + " " + str(f2._sweets))`
			5 20	Meaning is the same
				Both use
				*classes (Java **toya**) ,*
				*constructors (Java **toya** and Python `def __init__`),*
				objects (f1 and f2) and
				methods (maya and roko) to manipulate objects
				
				Minor differences, e.g Java has to have a main method while it's not mandatory in Python

Prepared by Ethel Tshukudu
Project: Programming Language Transfer
Centre for Computing Science Education/ University of Glasgow

Question 5

Java	Answer	Python	Answer	Does the Java code snippet mean the same as the Python code? Why?

`public class Main {`
`  public static void main(String[] args)`
`  {`
`  ArrayList num = new ArrayList(2);`
`  num.add(0);`
`  num.add(1);`
`  ArrayList numx=num;`
`  num.set(0, 2);`
`  System.out.println(num);`
`  System.out.println(numx);`
`  }}`
	[2,1][2,1]	`a=[0,1]`
		`b=a`
		`a[0]=2`
		`print(b)`
		`print(a)`
			[2,1][2,1]	Meaning is the same
				They both allow multiple references to the same object, which can them be changed through either reference variable.
				When a reference variable is assigned the value of another reference variable, e.g (b=a), the assignment makes the variable (b) contain a reference to the same object that variable a references – their values (address) are the equal. Results in
				E.g. #Python, any modifications to the object of (a) also apply to (b).
				# Java, any modifications to the object of (num) also apply to (numx).
				
				Differences e.g Java uses class ArrayList to offer methods similar to functionality of lists in Python.