SESSION - IV

Part - I

Lab No. 9: Generics

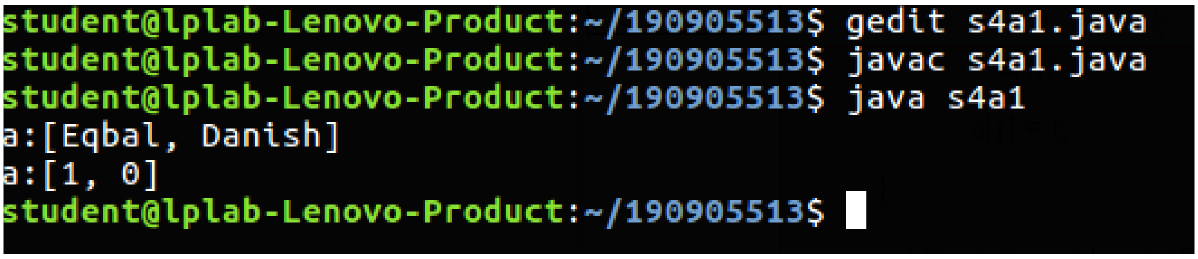
Lab Exercises

1. Write a generic method to exchange the positions of two different elements in an array.

Code:

```
import java.util.Arrays;
import java.util.List;
class s4a1
{
public static final <T> void swap (T[] a, int i, int j)
{
T t = a[i];
a[i] = a[j];
a[j] = t;
}
public static void main(String[] args)
{String [] a = {"Danish", "Eqbal"};
swap(a,0, 1);
System.out.println("a:"+Arrays.toString(a));
Integer [] b = {0, 1};
swap(b, 0, 1);
System.out.println("a:"+Arrays.toString(b));
}}
```

Test Case:



```
student@lplab-Lenovo-Product:~/190905513$ gedit s4a1.java
student@lplab-Lenovo-Product:~/190905513$ javac s4a1.java
student@lplab-Lenovo-Product:~/190905513$ java s4a1
a:[Eqbal, Danish]
a:[1, 0]
student@lplab-Lenovo-Product:~/190905513$
```

2. Define a simple generic stack class and show the use of the generic class for two different class types Student and Employee class objects.

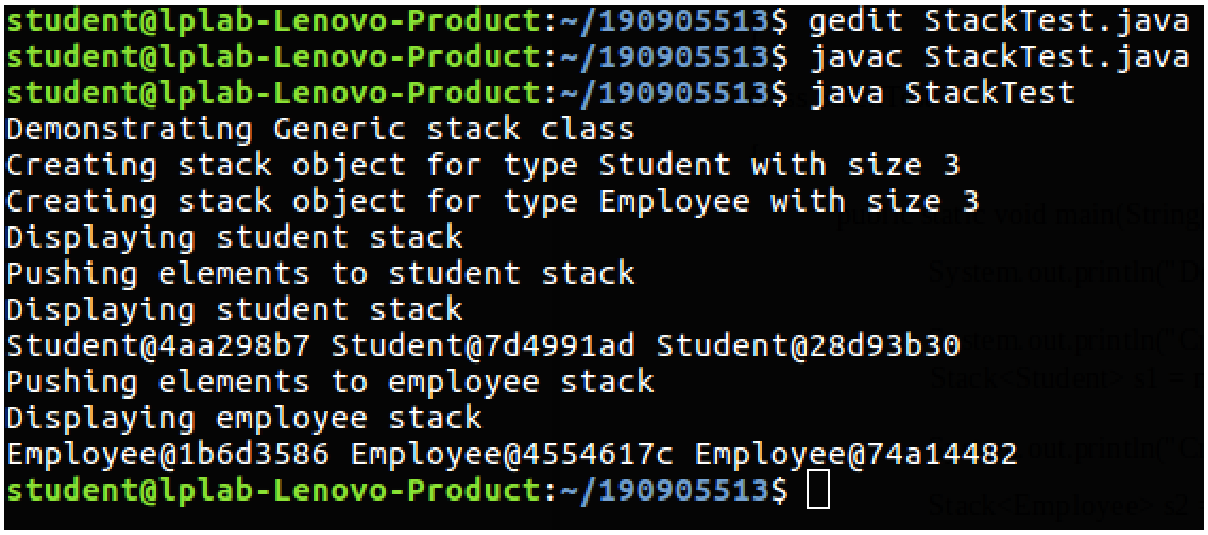
Code:

```
import java.lang.reflect.Array;

class PushException extends
Exception

{

private int code;

public PushException(int c)

{this.code = c;}

public int getCode()

{return code;}

}

class PopException extends
Exception

{

private int code;

public PopException(int c)

{this.code = c;}

public int getCode()

{return code;}

}
```

```
class Stack<T>

{

private T item[]; private int
top; private int size;

public Stack(Class<T[]> clazz,
int length)

{

this.size = length;

this.item =
clazz.cast(Array.newInstance(c
lazz.getComponentType(),
length));

this.top = -1;

}

public boolean isEmpty()

{

if(this.top == -1) return (true);

return (false);

}

public boolean isFull(){

if(this.top == this.size -1)
return (true);

return (false);

}

public boolean push(T elem)
throws PushException

{

if(this.isFull())
{ throw new
PushException(1);

}

this.item[++this.top] = elem;

return (true); }

public T pop() throws
PopException

{

if(this.isEmpty())

{throw new PopException(-1);

}

return(this.item[this.top--]);

}

public void display()

{

if(this.isEmpty()) return;

for(int i = 0; i < this.top + 1; i++)

{

System.out.print(this.item[i]);

System.out.print(" "); }

System.out.println("");

}

}

class Student

{

String name; int reg_no;
```

```
Student(String name, int
reg_no)

{ this.name = name;

this.reg_no = reg_no;

}

}

class Employee

{

String name; int emp_no;

Employee(String name, int
emp_no)

{ this.name = name;
this.emp_no = emp_no;

}

}

class StackTest

{

public static void
main(String[] args) {

System.out.println("Demonstra
ting Generic stack class");

System.out.println("Creating
stack object for type Student
with size 3");

Stack<Student> s1 = new
Stack<Student>(Student[].clas
s, 3);

System.out.println("Creating
stack object for type Employee
with size 3");
Stack<Employee> s2 = new
Stack<Employee>(Employee[]
.class, 3);

System.out.println("Displaying
student stack"); try

{

System.out.println("Pushing
elements to student stack");

s1.push(new Student("Danish",
12345));

s1.push(new Student("Nalin",
67890));

s1.push(new
Student("Abhinav", 13578));

System.out.println("Displaying
student stack");

s1.display();

System.out.println("Pushing
elements to employee stack");

s2.push(new
Employee("Srisai", 12568));

s2.push(new
Employee("Arora", 23579));

s2.push(new
Employee("Kenzel", 14795));

System.out.println("Displaying
employee stack");

s2.display();

}catch(PushException ex)

{System.out.println("Caught
push exception"); }

}}
```

Test Case:



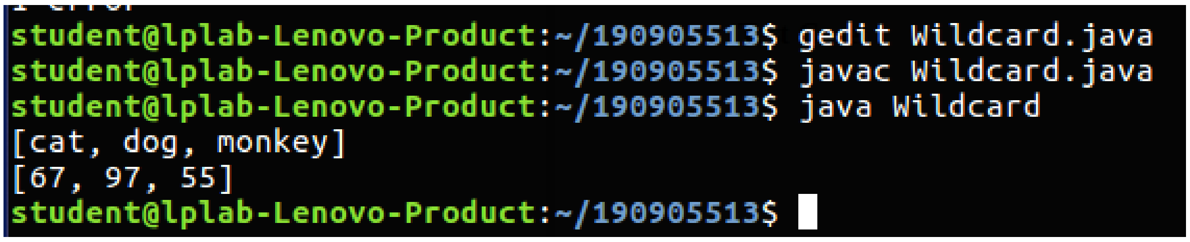
```
student@lplab-Lenovo-Product:~/190905513$ gedit StackTest.java
student@lplab-Lenovo-Product:~/190905513$ javac StackTest.java
student@lplab-Lenovo-Product:~/190905513$ java StackTest
Demonstrating Generic stack class
Creating stack object for type Student with size 3
Creating stack object for type Employee with size 3
Displaying student stack
Pushing elements to student stack
Displaying student stack
Student@4aa298b7 Student@7d4991ad Student@28d93b30
Pushing elements to employee stack
Displaying employee stack
Employee@1b6d3586 Employee@4554617c Employee@74a14482
student@lplab-Lenovo-Product:~/190905513$
```

3. Write a program to demonstrate the use of wildcard arguments.

Code:

```
import java.util.Arrays;

import java.util.List;

class Wildcard

{ private static void printlist(List<?> list)

{System.out.println(list); }

public static void main(String[] args)  {

List<String> list1= Arrays.asList("cat", "dog", "monkey");

List<Integer> list2=Arrays.asList(67, 97, 55);

printlist(list1);

printlist(list2);

} }
```

Test Case:



```
student@lplab-Lenovo-Product:~/190905513$ gedit Wildcard.java
student@lplab-Lenovo-Product:~/190905513$ javac Wildcard.java
student@lplab-Lenovo-Product:~/190905513$ java Wildcard
[cat, dog, monkey]
[67, 97, 55]
student@lplab-Lenovo-Product:~/190905513$
```