Session - III

Part I

Lab No. 7: Interfaces and Exception Handling

Lab Exercises

1. Design a stack class. Provide your own stack exceptions namely Push Exception and Pop Exception, which throw exceptions when the stack is full and when the stack is empty respectively. Show the usage of these exceptions in handling a stack object in the main.

Code:

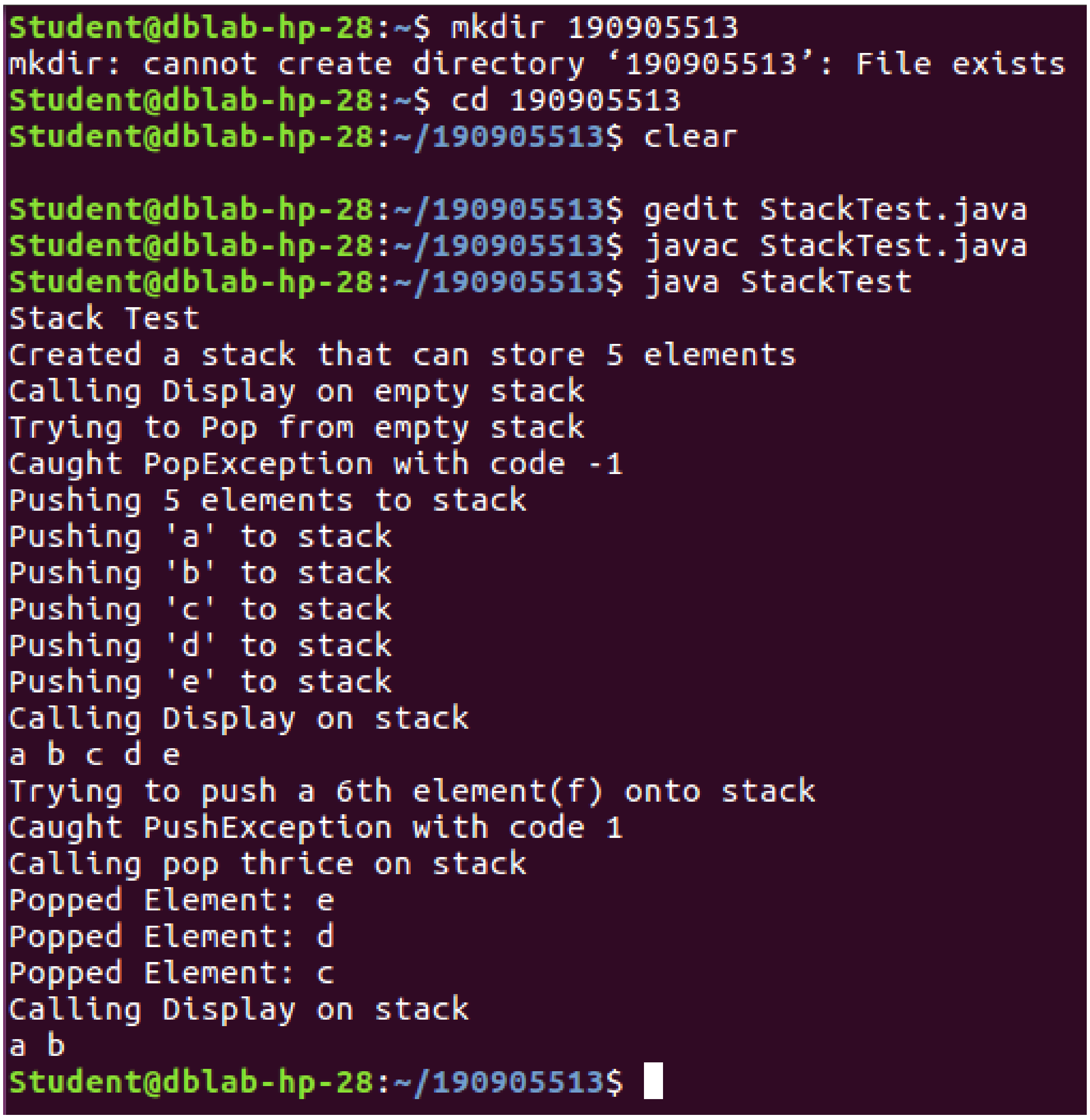
```
class PushException extends

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{
private char item[];
private int top;
private int size;
public Stack()
{ this.item = new char[0];
this.top = -1;
this.size =0;
}
public Stack(int size){
this.size = size;
this.item = new char[size];
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(char elem) throws PushException
{ if(this.isFull())
{ throw new PushException(1);
}
this.item[++this.top] = elem;
```

```
return (true);
}
public char 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(String.format("%c ", this.item[i]));
System.out.println("");
} }
class StackTest {
public static void main(String[] args) {
System.out.println( "Stack Test");
Stack s = new Stack(5);
System.out.println( "Created a stack that can store 5 elements");
System.out.println( "Calling Display on empty stack");
s.display();
System.out.println( "Trying to Pop from empty stack");
try{
char el = s.pop();
System.out.println("Popped element: " + el);
}
catch(PopException e){
System.out.print("Caught PopException with code ");
System.out.println(e.getCode());

}

System.out.println( "Pushing 5 elements to stack");

try{

System.out.println("Pushing 'a' to stack");

s.push('a');

System.out.println("Pushing 'b' to stack");

s.push('b');

System.out.println("Pushing 'c' to stack");

s.push('c');

System.out.println("Pushing 'd' to stack");

s.push('d');

System.out.println("Pushing 'e' to stack");

s.push('e');

System.out.println("Calling Display on stack");

s.display();

System.out.println("Trying to push a 6th element(f) onto stack");

s.push('f');

}

catch(PushException e){

System.out.print("Caught PushException with code ");

System.out.println(e.getCode());

}

System.out.println("Calling pop thrice on stack");

try{

System.out.println("Popped Element: " + s.pop());

System.out.println("Popped Element: " + s.pop());

System.out.println("Popped Element: " + s.pop());
```

```
}
catch(PopException e){
System.out.print("Caught PopException with code ");
System.out.println(e.getCode());
}
System.out.println("Calling Display on stack");
s.display();
}
}
```

Test Case:



```
Student@dblab-hp-28:~$ mkdir 190905513
mkdir: cannot create directory ‘190905513’: File exists
Student@dblab-hp-28:~$ cd 190905513
Student@dblab-hp-28:~/190905513$ clear

Student@dblab-hp-28:~/190905513$ gedit StackTest.java
Student@dblab-hp-28:~/190905513$ javac StackTest.java
Student@dblab-hp-28:~/190905513$ java StackTest
Stack Test
Created a stack that can store 5 elements
Calling Display on empty stack
Trying to Pop from empty stack
Caught PopException with code -1
Pushing 5 elements to stack
Pushing 'a' to stack
Pushing 'b' to stack
Pushing 'c' to stack
Pushing 'd' to stack
Pushing 'e' to stack
Calling Display on stack
a b c d e
Trying to push a 6th element(f) onto stack
Caught PushException with code 1
Calling pop thrice on stack
Popped Element: e
Popped Element: d
Popped Element: c
Calling Display on stack
a b
Student@dblab-hp-28:~/190905513$
```

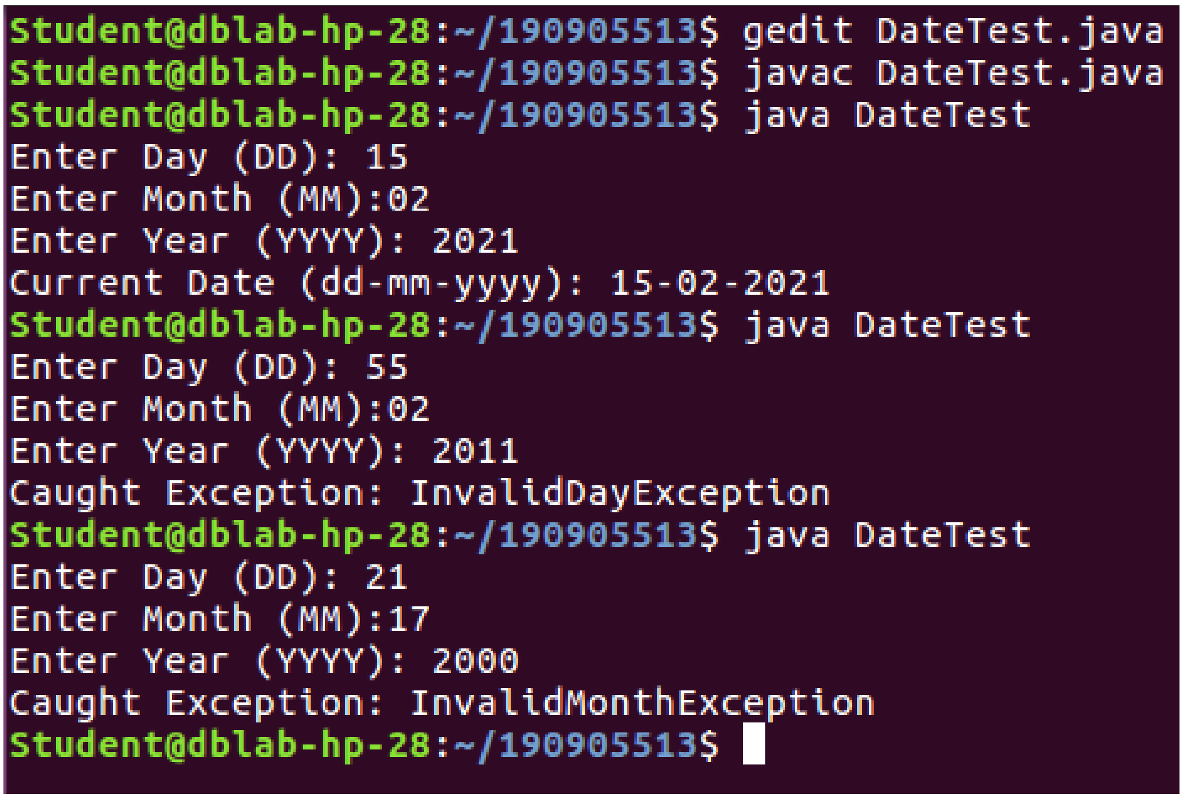
2. Define a class CurrentDate with data members day, month and year. Define a method createDate() to create date object by reading values from keyboard. Throw a user defined exception by name InvalidDayException if the day is invalid and InvalidMonthException if month is found invalid and display current date if the date is valid. Write a test program to illustrate the functionality.

Code:

```
import java.util.Scanner;
class InvalidDayException extends Exception{
int code;
public InvalidDayException(int c){
code = c;
}
public int
getCode(){
return code;
} }
class InvalidMonthException extends Exception
{int code;
public InvalidMonthException(int c){
code = c;
}
public int getCode(){
return code;
} }
class CurrentDate
{
private int day, month, year;
public CurrentDate()
{
this.day = 1;
this.month = 1;
this.year = 1991;
}
public CurrentDate(int day, int month, int year) throws InvalidDayException,
InvalidMonthException{
if(month > 12 || month < 1) throw new InvalidMonthException(month-12);
if(month==1||month==3||month==        5||month==7||month==        8||month==10||
month==12){
if(day > 31 || day < 1) throw new InvalidDayException(day-31);
}
if(month == 4||month == 6||month == 9||month == 11){
if(day > 30 || day < 1) throw new InvalidDayException(day-30);
}
if(month ==2){
if((year%4 == 0 && year%100 != 0) || year%400 == 0){
if(day > 29 || day < 1) throw new InvalidDayException(day-29);
}
else{
```

```
if(day > 28 || day < 1) throw new InvalidDayException(day-28);
}
}
this.day = day; this.month= month; this.year = year;
}
public void display(){
System.out.println(String.format("Current Date (dd-mm-yyyy): %02d-%02d-%04d", this.day, this.month, this.year));
}
}
class DateTest{
public static CurrentDate createDate() throws InvalidDayException, InvalidMonthException
{
Scanner sc = new Scanner(System.in);
System.out.print("Enter Day (DD): ");
int day = sc.nextInt();
sc.nextLine();
System.out.print("Enter Month (MM):");
int month = sc.nextInt();
sc.nextLine();
System.out.print("Enter Year (YYYY): ");
int year = sc.nextInt();
sc.nextLine();
try{CurrentDate d = new CurrentDate(day, month, year);
return d;
}
catch(InvalidDayException | InvalidMonthException ex){
throw ex;
}
}
public static void main(String[] args){
CurrentDate d;
try{
d =createDate();
d.display();
}
catch(InvalidDayException | InvalidMonthException ex){
System.out.print("Caught Exception: ");
System.out.println(ex);
}
}
}
```

Test Case:



```
Student@dblab-hp-28:~/190905513$ gedit DateTest.java
Student@dblab-hp-28:~/190905513$ javac DateTest.java
Student@dblab-hp-28:~/190905513$ java DateTest
Enter Day (DD): 15
Enter Month (MM):02
Enter Year (YYYY): 2021
Current Date (dd-mm-yyyy): 15-02-2021
Student@dblab-hp-28:~/190905513$ java DateTest
Enter Day (DD): 55
Enter Month (MM):02
Enter Year (YYYY): 2011
Caught Exception: InvalidDayException
Student@dblab-hp-28:~/190905513$ java DateTest
Enter Day (DD): 21
Enter Month (MM):17
Enter Year (YYYY): 2000
Caught Exception: InvalidMonthException
Student@dblab-hp-28:~/190905513$
```

3. Design a Student class with appropriate data members as in Lab 5. Provide your own exceptions namely Seats Filled Exception, which is thrown when Student registration number is >XX25 (where XX is last two digits of the year of joining) Show the usage of this exception handling using Student objects in the main. (Note: Registration number must be a unique number).

Code:

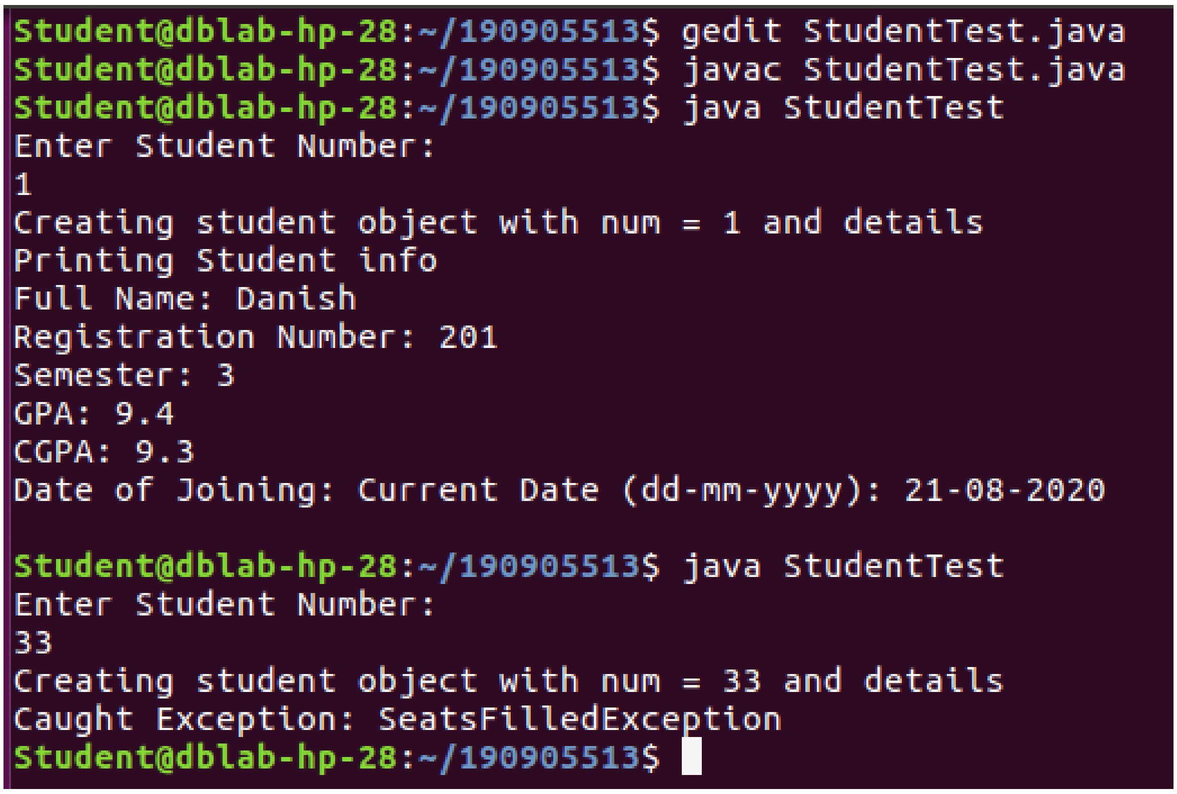
```
import java.util.Scanner; class

InvalidDayException

extends Exception{

int code;

public InvalidDayException(int c){

code = c;

}

public int
```

```
getCode(){

return code;

}

}

class InvalidMonthException extends Exception{

int code; public

InvalidMonthException(int c) {

code=c;

}

public int getCode(){

return code;

}

}

class SeatsFilledException extends Exception{

int code;

public SeatsFilledException(int c){

code = c;

}

public int getCode(){

return code;

}

}

class Date

{

int day, month, year;

public Date()

{
```

```
this.day = 1;
this.month = 1;
this.year = 1991;
}
public Date(int day, int month, int year) throws InvalidDayException,
InvalidMonthException{
if(month > 12 || month < 1) throw new InvalidMonthException(month-12);
if(month== 1||month == 3||month == 5||month== 7||month == 8||month==10||
month== 12){
if(day > 31 || day < 1) throw new InvalidDayException(day-31);
}
if(month == 4||month == 6||month == 9||month == 11){
if(day > 30 || day < 1) throw new InvalidDayException(day-30);
}
if(month == 2){
if((year%4 == 0 && year%100 != 0) || year%400 == 0){
if(day > 29 || day < 1) throw new InvalidDayException(day-29);
}
else
{
System.out.println(day);
if(day >28 || day < 1) throw new InvalidDayException(day-28);
}
}
this.day = day;
this.month = month;
this.year = year;
```

```
}
public String getDate(){
return(String.format("Current Date (dd-mm-yyyy): %02d-%02d-%04d", this.day,
this.month, this.year));
}
}
class Student
{
private int regNo;
private String fullName;
private Date dateJoining;
private short semester;
private float gpa; private
float cgpa;
public Student(String fullName, Date dateJoining, short semester, float gpa, float
cgpa, int num) throws SeatsFilledException{
if(num > 25) throw new SeatsFilledException(num);
this.fullName = fullName;
this.dateJoining = dateJoining;
this.semester = semester;
this.gpa = gpa; this.cgpa =cgpa;
String reg_year = String.format("%04d", this.dateJoining.year);
String reg = reg_year.substring(2, 4) + String.format("%s", num);
this.regNo = Integer.parseInt(reg);
}
public Student(){
this.fullName = "";
this.dateJoining= new Date();
this.semester =0;

this.gpa = 0;

this.cgpa =0;

this.regNo = 0;

}

public void printStudentInfo(){

System.out.println ("Full Name: " + this.fullName);

System.out.println ("Registration Number: " + this.regNo);

System.out.println ("Semester: " + this.semester);

System.out.println ("GPA: " + this.gpa);

System.out.println ("CGPA: " + this.cgpa);

System.out.println ("Date of Joining: " + this.dateJoining.getDate());

System.out.println ("");

}

}

class StudentTest{

public static void main(String[] args){

Scanner sc = new Scanner(System.in);

try{

Date doj1 = new Date(21, 8, 2020);

System.out.println("Enter Student Number:");

int num = sc.nextInt();

sc.nextLine();

System.out.println(String.format("Creating student object with num = %d and
details", num));

Student s = new Student("Danish", doj1, (short) 3, 9.4f, 9.3f,

num); System.out.println("Printing Student info");
s.printStudentInfo();

}

catch(InvalidDayException | InvalidMonthException | SeatsFilledException ex){

System.out.print("Caught Exception: ");

System.out.println(ex);

}

} }
```

Test Case:



```
Student@dblab-hp-28:~/190905513$ gedit StudentTest.java
Student@dblab-hp-28:~/190905513$ javac StudentTest.java
Student@dblab-hp-28:~/190905513$ java StudentTest
Enter Student Number:
1
Creating student object with num = 1 and details
Printing Student info
Full Name: Danish
Registration Number: 201
Semester: 3
GPA: 9.4
CGPA: 9.3
Date of Joining: Current Date (dd-mm-yyyy): 21-08-2020

Student@dblab-hp-28:~/190905513$ java StudentTest
Enter Student Number:
33
Creating student object with num = 33 and details
Caught Exception: SeatsFilledException
Student@dblab-hp-28:~/190905513$
```