Week 5

1)

```
class PushException extends Exception{
	private int code;

	PushException(int c){
		this.code = c;
	}

	int getCode(){
		return code;
	}
}

class PopException extends Exception{
	private int code;

	PopException(int c){
		this.code = c;
	}

	int getCode(){
		return code;
	}
}
class Stack{
	char a[];
	int top;
	int size;

	Stack(){
		a = new char[0];
		top = -1;
		size = 0;
	}
	Stack(int s){
		a = new char[s];
		top = -1;
		size = s;
	}

	boolean isEmpty(){
		if(top == -1)
			return true;
		return false;
	}
	boolean isFull(){
		if(top == size-1)
			return true;
		return false;
```

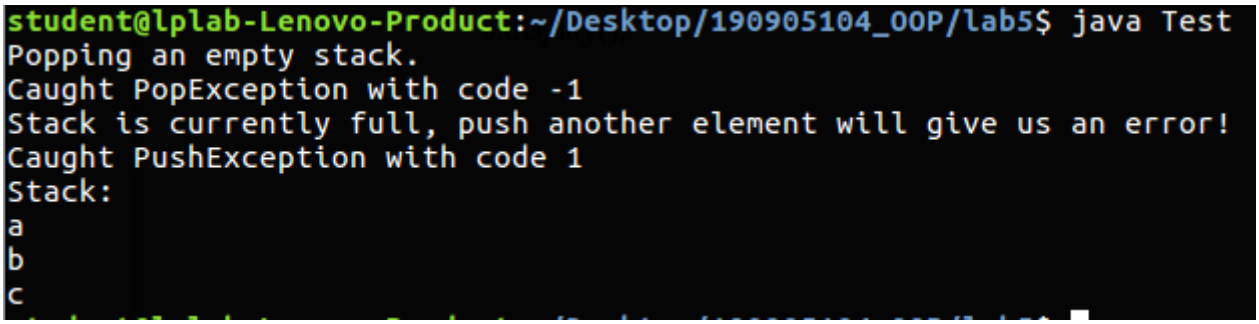
```
	}

	boolean push(char c) throws PushException{
		if(isFull()){
			throw new PushException(1);
		}
		a[++top] = c;
		return (true);
	}
	char pop() throws PopException{
		if(isEmpty()){
			throw new PopException(-1);
		}
		return(a[top--]);
	}

	void display(){
		if(isEmpty())
			return;
		System.out.println("Stack: ");
		for(int i = 0; i < top+1; i++)
			System.out.println(a[i]);
	}
}

class Test{
	public static void main(String []args){
		Stack s = new Stack(3);
		s.display();
		System.out.println("Popping an 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());
		}
		try{
			s.push('a');
		}catch(PushException e){
			System.out.print("Caught PushException with code ");
			System.out.println(e.getCode());
		}
		try{
			s.push('b');
		}catch(PushException e){
			System.out.print("Caught PushException with code ");
			System.out.println(e.getCode());
		}
		try{
			s.push('c');
		}catch(PushException e){
```

```
                System.out.print("Caught PushException with code ");
                System.out.println(e.getCode());
            }
            System.out.println("Stack is currently full, push another element will give us an
error!");
            try{
                s.push('d');
            }catch(PushException e){
                System.out.print("Caught PushException with code ");
                System.out.println(e.getCode());
            }
            s.display();
        }
}
```



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.
import java.util.Scanner;

class InvalidDayException extends Exception{
        private int code;

        InvalidDayException(int c){
            code = c;
        }

        int getCode(){
            return code;
        }
}

class InvalidMonthException extends Exception{
        private int code;

        InvalidMonthException(int c){
            code = c;
```

```
	}

	int getCode(){
		return code;
	}
}

class CurrentDate{
	int date, month, year;

	void createDate(){
		date = 1;
		month = 1;
		year = 2000;
	}

	void createDate(int d, int m, int y) throws InvalidDayException, InvalidMonthException{
		if(m <1 || m >12)
			throw new InvalidMonthException(-1);
		month = m;
		switch(month){
			case 1:
			case 3:
			case 5:
			case 7:
			case 8:
			case 10:
			case 12:
			if(d < 1 || d > 31)
				throw new InvalidDayException(-1);
			break;
			case 4:
			case 6:
			case 9:
			case 11:
			if(d < 1 || d > 30)
				throw new InvalidDayException(-1);
			break;
			case 2:
			if(d < 1 || d > 28)
				throw new InvalidDayException(-1);
		}
		date = d;
		year = y;
	}

	public void display(){
		System.out.println(String.format("Current Date (dd-mm-yyyy): %02d-%02d-%04d",
date, month, year));
	}
}
```

```
class DateTest
{
	public static CurrentDate createDate() throws InvalidDayException, InvalidMonthException
	{
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter dd mm yyyy");
		int d = sc.nextInt();
		int m = sc.nextInt();
		int y = sc.nextInt();
		try{
			CurrentDate date = new CurrentDate();
			date.createDate(d, m, y);
			return date;
		}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);
		}
	}
}
```

```
student@lplab-Lenovo-Product:~/Desktop/190905104_OOP/lab5$ java DateTest
Enter dd mm yyyy
28 5 2001
Current Date (dd-mm-yyyy): 28-05-2001
student@lplab-Lenovo-Product:~/Desktop/190905104_OOP/lab5$ java DateTest
Enter dd mm yyyy
32 5 2001
Caught Exception: InvalidDayException
```

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)

import java.util.Scanner;
```

```
class SeatsFilledException extends Exception{
        private int code;

        SeatsFilledException(int c){
                code = c;
        }
        int getcode(){
                return code;
        }
}

class Student{
         int reg_no;
         int date, month, year;
         String name;
         short sem;
         float gpa, cgpa;

         Student(){
                  name = "";
                  reg_no = 0;
                  date = 0;
                  month = 0;
                  year = 0;
                  sem = 0;
                  gpa = 0;
                  cgpa = 0;
         }

         Student(String s, int d, int m, int y, int reg, short sem_no, float gp, float cg)throws
SeatsFilledException{
      name = s;
      date = d;
      month = m;
      year = y;
      if(reg % 100 > 25)
          throw new SeatsFilledException(-1);
      reg_no = reg;
      sem = sem_no;
      gpa = gp;
      cgpa = cg;
     }
}

class StudentTest{
        public static void main(String[] args){
                Scanner sc = new Scanner(System.in);
                float gp,cg;
                int d, m, y, reg;
                short sem;
                String name = "";
```

```
            System.out.println("Enter name");
            name = sc.nextLine();
            System.out.println("Enter date of joining(dd mm yyyy)");
            d = sc.nextInt();
            m = sc.nextInt();
            y = sc.nextInt();
            System.out.println("Enter registration number");
            reg = sc.nextInt();
            System.out.println("Enter GPA");
            gp= sc.nextFloat();
            System.out.println("Enter CGPA");
            cg = sc.nextFloat();
            System.out.println("Enter semester");
            sem = sc.nextShort();

            try{
                Student stu = new Student(name, d, m, y, reg, sem, gp, cg);
            }
            catch(SeatsFilledException e){
                System.out.println("Seats are filled. Try again next year.");
            }
        }
}
```

```
student@lplab-Lenovo-Product:~/Desktop/190905104_OOP/lab5$ java StudentTest
Enter name
Parth
Enter date of joining(dd mm yyyy)
28 5 2019
Enter registration number
1913
Enter GPA
9
Enter CGPA
9
Enter semester
3
student@lplab-Lenovo-Product:~/Desktop/190905104_OOP/lab5$ java StudentTest
Enter name
Jhgyhj
Enter date of joining(dd mm yyyy)
18 4 2019
Enter registration number
1975
Enter GPA
6
Enter CGPA
6
Enter semester
3
Seats are filled. Try again next year.
```