Exceptions & Exception Handling

Example: Java

```
public class Crash {
    static int a = {10, 20, 30, 40, 50};
    public static void main (String[] args) {
        for (int i=0; i < a.length; i++)
            System.out.printf ("%d\n", a[i]);
        System.out.printf ("Done printing the array!\n");
    } //main()
} //class Crash
```

Change loop condition to: i <= a.length

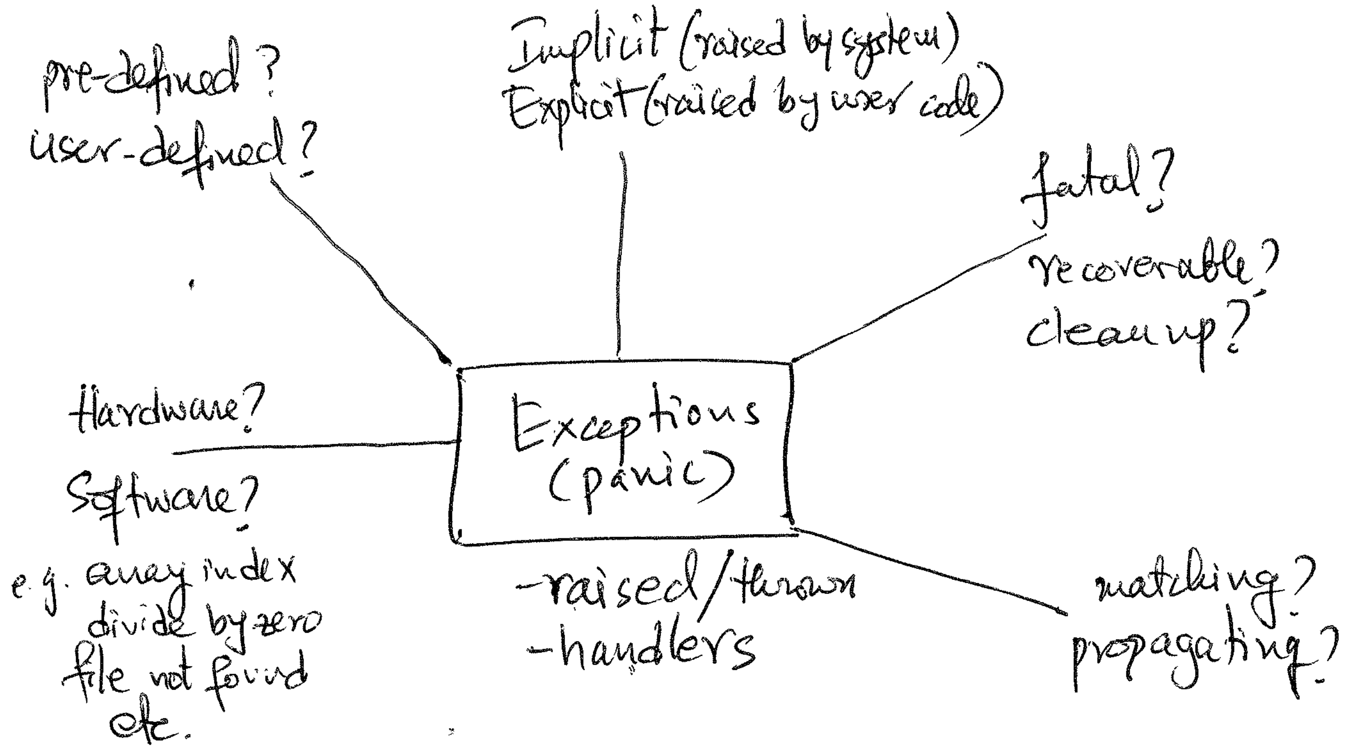
to get an exception: java.lang.ArrayIndexOutOfBoundsException

quits the program.

We can write an Exception Handler to address the issue.

Example: Java Exception Handler

```
import java.util.Scanner;
public class Crash2 {

    public static void main(String[] args) {

        int n;

        Scanner s = new Scanner(system.in);

        while (true) {

            try {
                system.out.printf("Enter an int :");
                n = Integer.parseInt(s.next());
                system.out.printf("%d\n", n);
            } catch (java.lang.NumberFormatException e) {
                System.out.printf("Error: ...");
            }

        }
    } //main()
} //class Crash2
```



Exception Handlers

① Recover and continue
② Cannot recover, but require clean up
③ Nothing can be done, exit gracefully by printing message

Questions: - Does the PL support exceptions?
- How are exception handlers specified?
- What is their scope?
- Can information about the exception be passed to handler?
- Does the PL allow a continuation or a termination model?

In general

```
try:
    ≡
except <e1>:
    ≡
except <e2>:
    ≡
finally:        ← always executed
    ≡
else:           ← enters when no exception
    ≡             is raised in try
```

Raising an exception

```
raise <e>
```

User-Defined Exceptions

```
class StackError (Exception):
    def __init__(self, msg="StackError")
        self.message = msg
        super.__init__(self.message)
```

```
def pop(self):
    if (self.isEmpty()):
        raise StackError("Stack Underflow")
    ≡
```

use

```
try:
    ≡
    item = s.pop()
    |
except StackError, e:
    ≡
```

$\overset{as?}{,}$

Exceptions in Java

(i) Review EH in Java handout.

- checked exceptions: conditions that may be outside program control
  - checked a compile time
  - require try-catch block.

- unchecked exceptions
  - not checked at ~~runtime~~ compile time
  - may occur during runtime (e.g. ArithmeticException)

Handlers

```
try {
      ≡
 } catch (<exception type> e1) {
      ≡
  } catch (<exception type> e2) {
      ≡
   }
   :
   } finally {
        ≡ always executed if present
     }
```

Exception Methods

- String getMessage()
- void printStackTrace()

Java: User-defined Exceptions

In stack implementation, an underflow: trying to pop ~~[illegible]~~ an empty stack...

```
public class stackError extends Exception {
    public stackError (String message){
            super(message);
    }
} //stackError
```

```
public E pop () throws stackError {
    ===
    if (empty())
        throw new StackError("Stack Underflow");
}//pop()
```

Use

```
try {
    - - - -
    ~~pop~~ item = s.pop();
        ==
}catch (stackError e) {
    ==
}
```

Exceptions in Python

review Handout

```
ns = "53"
name = "Deepak"
```

① print(int(ns)) $\longrightarrow$ 53
print(int(name)) $\longrightarrow$ ValueError: - - . .

② 5 + "Deepak" $\longrightarrow$ TypeError

③ 5/0 $\longrightarrow$ ZeroDivisionError

④ d = {'a' : 1, 'b' : 2}
d['a'] $\longrightarrow$ 1
d['c'] $\longrightarrow$ KeyError

⑤ l = [10, 20, 30]

```
try:
    print(l[0])
    print(l[3])   # err.
except:
    print("Error - -.")
```