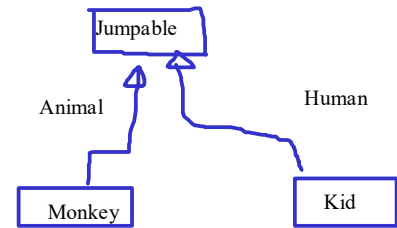
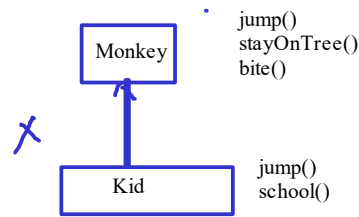
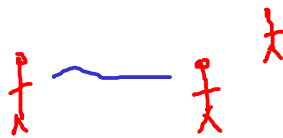


Coding convention:  
 name of class should start with capital letter  
 ✓ PartTimeEmployee  
 ✗ parttime\_employee

data  
 payPerHr

When to use what?  
 abs class vs interface



Strings are immutable (once it created there state can not be change)

How String stored in java?

✓ String s1=new String("lee");  
 or  
 ✓ String s2="lee";

String s3="lee"

s3=s3+"foo"

Bad prog practice

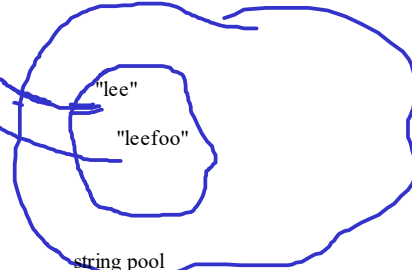
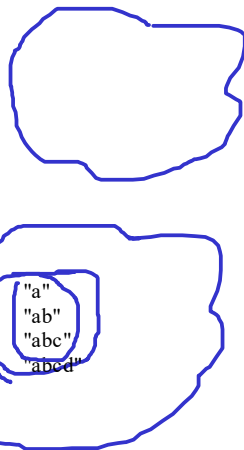
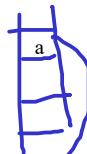
String a="a"+"b"+"c"+"d";

stack

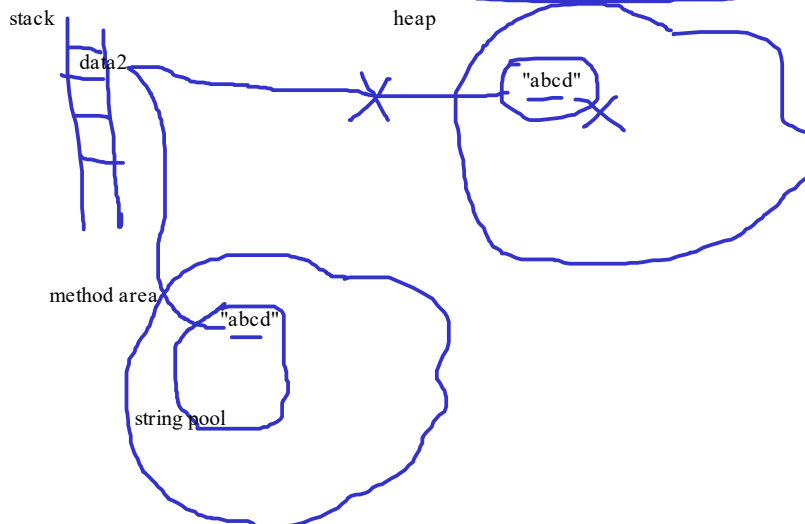
heap

method area

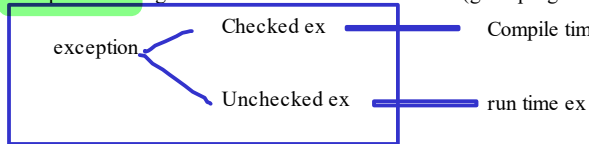
string pool



✓ String data2=new StringBuilder().append("a").append("b").append("c").append("d").toString();



# exception handling: Checked and unchecked ex? GPP(good programming practice)



All Ex happend at run time?

try  
catch  
throw  
throws  
finally

"God give me the problem  
but give me strenght to handle that"

"robust"

code

Compiler

Examples: divide method



```
try {
    int x, y;
    Scanner scanner=new Scanner(System.in);
    System.out.println("PE 2 nos");
    x=scanner.nextInt();
    y=scanner.nextInt();

    int z=x/y;
    System.out.println("Z:"+z);

    System.out.println("end");
}
catch(ArithmeticException e) {
    System.out.println("dont do divide by zero");
}
catch(InputMismatchException e) {
    System.out.println("input must be ints");
}
```

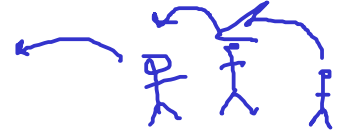
DRY

```
Scanner scanner = null;
try {
    int x, y;
    scanner = new Scanner(System.in);

    System.out.println("PE 2 nos");
    x = scanner.nextInt();
    y = scanner.nextInt();

    int z = x / y;
    System.out.println("Z: " + z);

    System.out.println("end");
    scanner.close();
} catch (ArithmeticException e) {
    System.out.println("dont do divide by zero");
}
catch (InputMismatchException e) {
    System.out.println("input must be ints");
}
```



### User define exception:

throw vs throws

try

catch

throw

throws

finally

Need of user define ex?

bankapp

Account

SA

withdraw

1000

deposit

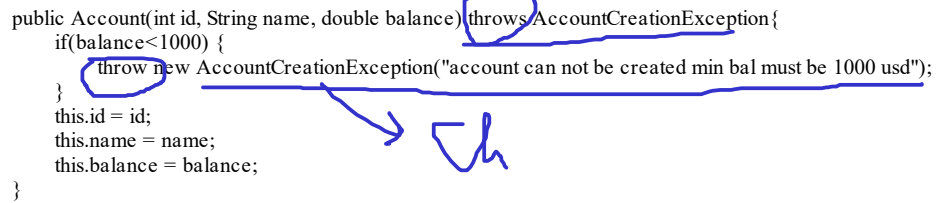
account creation process

NotSufficientFundException

OverFundException

AccountCreationException

```
public Account(int id, String name, double balance) throws AccountCreationException {  
    if(balance < 1000) {  
        throw new AccountCreationException("account can not be created min bal must be 1000 usd");  
    }  
    this.id = id;  
    this.name = name;  
    this.balance = balance;  
}
```



```

class AccountCreationException extends Exception{
    private static final long serialVersionUID = 1825616456008488982L;

    public AccountCreationException(String message) {
        super(message);
    }
}

```

```

//NotSufficientFundException
class NotSufficientFundException extends Exception{
    private static final long serialVersionUID = 1825616456008488982L;

    public NotSufficientFundException(String message) {
        super(message);
    }
}

```

```

//OverFundException
class OverFundException extends RuntimeException{
    private static final long serialVersionUID = 1825616456008488982L;

    public OverFundException(String message) {
        super(message);
    }
}

```

```

class Account{
    private int id;
    private String name;
    private double balance;

    public Account(int id, String name, double balance) throws AccountCreationException{
        if(balance<1000) {
            throw new AccountCreationException("account can not be created min bal must be 1000 usd");
        }
        this.id = id;
        this.name = name;
        this.balance = balance;
    }

    public void withdraw(double amount) throws NotSufficientFundException {
        double temp=balance-amount;
        if(temp<1000) {
            throw new NotSufficientFundException("transaction is not possible min bal should be 1000 usd");
        }
        else
            balance=temp;
    }
    //IL
    public void deposit(double amount) throws OverFundException {
        double temp=balance+amount;
        if(temp>=10_00_00) {
            throw new OverFundException("transaction is not possible max bal should be less then 10_00_00 usd");
        }
        else
            balance=temp;
    }
    public void print() {
        System.out.println("id: "+ id +" name: "+ name +" balance: "+balance);
    }
}

public class A_UserDefineEx {

```

```

    public static void main(String[] args) {
        Account account;
        try {
            account = new Account(1, "pol", 900);
            account.depsit(5000);
            account.withdraw(94500);

            account.print();
        } catch (AccountCreationException e) {
            System.out.println(e.getMessage());
        } catch (OverFundException e) {
            System.out.println(e.getMessage());
        } catch (NotSufficientFundException e) {
            System.out.println(e.getMessage());
        }

        System.out.println("code end");
    }
}

```

BikeStoreApplication

min bike should be 10

sell(int qty)

procure(int qty)

BikeStoreCreationException

BikeNotAvailabeException

StoreFullException 20

BikeStore

BikeStoreDemo  
main