07/08/2020 by yosef

Java - OOP
.summary.

* Java don't have:
  * Pointers arithmatic.
  * Preprocessor
  * automatic type conversion.
  * Global function and varibales
  * operator overLoading.
  * multi inheritance.

* Java offers more GC algorithms.
* Java GC:

ה GC ב Java עובד כאשר מתמלא ה heap ונכנס לפעולה
אז הוא מנסה לנקות את ה objects ה unreachable.

Java -Class:

example:

```
public class ship
{
    Private String shipcap;

    public void setcaptainName(string name)
        { captainname = name;}
}
```

* if we want to give outside access we should do get/set

Default constructors

- Every class has a constructor, in case the programmer didn't write any constructor, the default will be supplied

* Java classes may be organized into packages.
* Every such package should contain classes that all related to the same ~~object~~ subject

float - 32 bit
double - 64 bit
} default float is part of double.

= default.

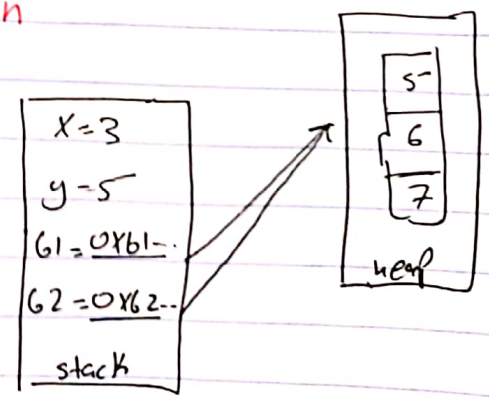
```
public class Example
{
    private int counter = 0; // counter is a primitive
    private Box b1; // b1 is a null reference.
    :
```

Look at the constructor

```
public Example(..., ..., ...)
{
    :
    b1 = new Box(..., ..., ...);
    :
}
```

^Reference in action^

```
int x = 3, y = 5;
Box b1 = new Box(5,6,7);
Box b1 = b2;
```



* The only way to pass arguments at Java is by value.

Arrays:

* memory for the array reference is allocated at the stack

* memory for the array object is allocated dynamically on the heap (Java)

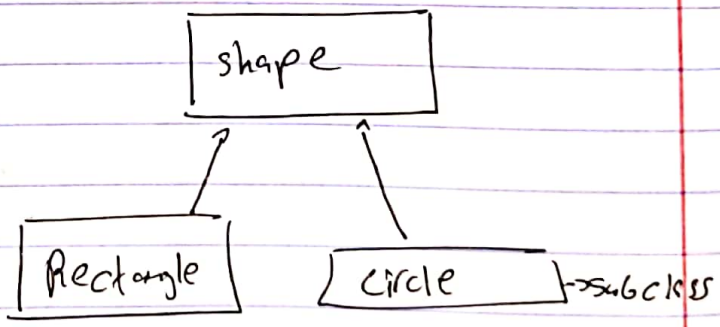
* to copy an array we can use System.arraycopy(sourceArr, src_st_indx, target, target_indx, length)

inheritance:

* Java implement single inheritance

```
public class shape {
    protected double area;
    public double getArea() {...}
}

public class circle extend shape {
    ⋮
}
```



* Attributes of the subclass can hide members of the superclass. in this case we use super.

* static members (neither variables nor methods) are never inherited.

Casting

* use instanceof to test the type of an object.

- The equals method -

* the operator == determines if two refrence are identical to each other (that is refer to the same object.)

* the equals method determines if the object are equal by there contents, but nessarily have the same refrence.

* static Keyword *

* are shared among all instance of classes
* static attribute can be accessed from outside the class if marked as public.
* static block code (static initializer or static constructor) execute only once, when the class is loaded.

The Singelton :: Design Pattern

* may be instantiated only once.

* the client shouldn't be able to instantiate it
  - private constructor
  - static variable

- the class let user to get that only instance
  static class method.

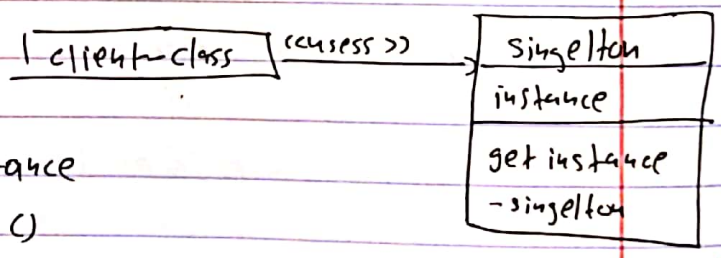
example:

```
public class Company {

    private static Company instance
                = new Company()

    private Company() {...}

    public static Company getCompany()
    {
        return instances;
    }
```



* client usage: ↙↙ client site

```
Company c = company.getCompany();
```

^The final keyword summary^

- you can't subclass a final class
- you can't override a final method

- A final variable is a constant.

Abstract class

- a class that can not be instaticated
- a class that declearel as abstract:

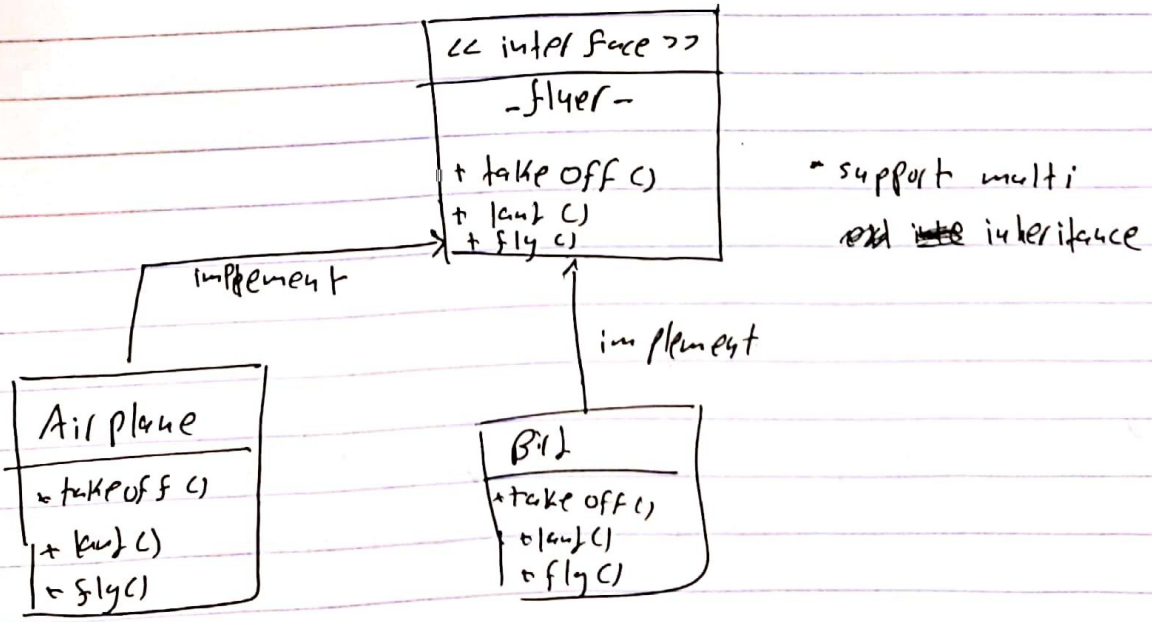
```
public abstract class shape{ _ }
    public abstract void paint(); // no body.
```

- Can't be both final & abstract at the same time.

interfaces

- A "public interface" is a contract between client code and the class that implement that interface

- A java interface is a formal decleration of such contract in which all method contains no implemenation



- support multi ~~extend~~ ~~inte~~ inheritance

Exceptions & Errors

Error:

* Used to indicated problems that mostly can't be fixed in Runtime

Like,

stack overflow Error
out of memory Error

Exceptions:

there are two kinds of Exceptions:

1- run time Exception:

- any expection that extend Runetime exception
-
- counted as Bugs and must be fixed to complete App.
- unchecked by the compiler - developer responsibility.

2- Application Exception:

- any expection that doesn't extend RunetimeExpection

- user defined exceptions.

```
try {
    :
    // Block where we expect to throw
    // a expection
}
catch (....) {
    :
}
```

throws - declears all Thrown expections
throw - actually creates an expection & throw it.

example:

```
public class check
{
    public static int check (string s) throw NFormatExce {
        int x = Integer.parseInt (s);
        if (x > 100)
            throw new NFormatExce ("number is Big");
        return x;
    }
```

Java collection

- Java object that holds an object group.
- number of objects in the group is dynamic

- there are 4 types of collection:

- collection: unordered group, duplicates are permitted
- set - unordered group, duplicates are forbidden.
- List - ordered group, ...
- map - group of Key - value pairs

Implement

Vector, ArrayList, LinkedList

Vector: synchronized

ArrayList, LinkedList: not synchronized

comparator: specifies how to compare between two objects
comparable: // // an object is compared to another.