Ch 1

Double Cast:
(double) a/b ✓
a/(double) b ✓
(double)(a/b) ×

Int-Max: $2^{31}-1$
Int-Min: -2^{31}

Escape Seq.
\n
\"
\\

Loops
- SCOPE

*Round-off error
↳ doubles cannot be represented exactly in binary

Integer division truncates

DeMorgan: $\neg(A \wedge B) = \neg A \vee \neg B$
$\neg(A \vee B) = \neg A \wedge \neg B$

Distributive: $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$

Other: $A \wedge A \equiv A$ $A \wedge T \equiv A$ $T \vee A \equiv$ True
$A \vee A \equiv A$ $F \vee A \equiv A$
$\neg(\neg A) \equiv A$ $F \wedge A \equiv$ False

Operator Precedence
highest ① !, ++, --
② *, /, %
③ +, -
④ <, >, <=, >=
⑤ ==, !=
⑥ &&
⑦ ||
lowest ⑧ =, +=, -=, *=, /=, %=

Short-Circuit Eval:
↳ Java stops evaluating a test if it knows the answer.

Random: int n = (int)(Math.random() * k) + p
[p, p+k) or [p, p+k-1]

Ch 2 - Classes & Objs

Static field - shared by all instances
overloading - same name different params
this - implicit parameter

PARAMETERS
Reference - objects, arrays, arraylists
Copy - primitive, Strings

Setting objs equal means sharing their memory.
• An uninitialized object is a null reference

Ch 3 - Inheritance, Polymorphism

keyword: extends → to a base class
↳ Subclasses inherit public or protected variables and methods. It can override methods.
→ Partial overriding: super() goes up one level
super(m) ← constructor. has to be at top
super.method(m); ← method.
→ CONSTRUCTORS ARE NOT INHERITED
- If no constructor is written for a subclass, the superclass default constructor is generated. If the superclass does not have a default (zero args) constructor, a compile error occurs.

```
public SubClass(){
    super();
}
```

It does not inherit private but CAN use ACCESSOR & MUTATOR methods

POLYMORPHISM
Higher Thing = Lower Thing
Only includes methods of Higher Thing but the overridden ones of Lower Thing.
To get Lower Thing methods, DOWNCAST
((Lower Thing) Varname)
↳ what method? Go to LOWEST overriden one.

If bad cast, classCastException

```
method (superclass){
}
```

OK: UPCAST

CALL:
method(subclass) ✓

ABSTRACT CLASSES

- Superclass representing abstract concept
- should not be instantiated
- May contain abstract methods ← no implementation, only header
- If a class contains any abstract methods it MUST be abstract
→ Can have some implemented & abs. methods

```
public abstract class Shape{
    private String name;
    public Shape(String shapeName){
        name = shapeName;
    }
    public String getName(){return name;}
    public abstract double area();
    public abstract double perimeter();
}
```

- Can have both instance & concrete methods
- A concrete subclass of an abstract superclass must provide implementation for all abstract methods otherwise it must also be declared abstract.
- YOU CANNOT CREATE INSTANCES

INTERFACES

Interfaces

- A collection of related methods that are abstract(tested) or default(Java 8, not tested) (implicitly)
- Non-default methods: PUBLIC & ABSTRACT
- Represents unrelated classes
- A class that implements an interface can define any amt of methods. Has to provide implementations for ALL abstract methods. If it fails to do this, the class must be declared abstract.

```
public interface FlyingObject{
    void fly();                  //abstract
    boolean isFlying();          // & public
}

public class Bird implements FlyingObject{
    ...
}
```

* extends precedes implements

- Interface - NO INSTANCE VARS
- Abstract Class - Can have instance vars.

Static/early binding - overload (is legal?)
Dynamic/late binding - override (which one?)

Ch 4 - Some Standard Classes

Object - superclass of everything
String - toString() returns a String, used in print, override this
- .equals(w) true or false == compares MEMORY by default
- immutable objects, Use +
- .compareTo(w)

JAVA QUICK REFERENCE USE IT

Wrapper Classes (ALSO IMMUTABLE)

Integer	Double
Integer(int v)	Double(double v)
.compareTo	
.intValue()	.doubleValue()
.equals(Object obj) (has to be valid Integer)	
.toString()	

Math
- See Random on prev. page
- JAVA QUICK REF.

ArrayLists
List <<Interface>>
ArrayList
List = ArrayList polymorphism
Use WRAPPERS
Boxing/Unboxing (Auto)
List<E> Methods:
- .add(w), (m,w)
- .size()
- .get(w)
- .set(m,w)
- .remove(w)

JAVA QUICK REFERENCE

2D
[rows][cols]

Ch 6 - Arrays & ArrayLists

type[] hi = new type[SIZE]; YOU NEED TO USE new unless
Initializer List — w = {1,2,3,4};

Arrays
Length - .length (Strings-.length(), ALists-.size())
* Do not use for-each when removing/replacing
* Arrays are passed in by MEMORY

2D GRID
call (row+1, col)
call (row-1, col)
call (row, col+1)
call (row, col-1)

Ch 7 - Recursion

- Calls itself
- base case (terminating condition) & body/recursive call

HELPER

```
int sum(int n){
    if (n==1) return 1;
    else return n+sum(n-1);
}
```

DRIVER

```
int getSum(int n){
    if (n>0) return sum(n);
    else {
        throw new Exception("n");
    }
```

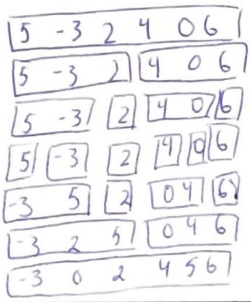
Ch 8 - Sorting & Searching

Selection Sort - "search & swap" ($O(n^2)$)
- Find the smallest element & swap with "top"
- When two elements left (a[n-2] and a[n-1]), place smaller one in a[n-2] and larger one in a[n-1]. The sort is complete
- For n elements, array is sorted after n-1 passes. After the kth pass, the first k elements are in their final sorted position.

Insertion Sort - ($O(n^2)$) two parts: sorted & unsorted
- Move elements from unsorted list to the sorted one; as each item is moved, it is inserted into its correct position in the sorted list (move down elements to create a slot)

Merge Sort $O(n \log n)$
1. If there is more than one elem. in array:
   - a. Break arr into 2 halves
   - b. Mergesort left
   - c. Mergesort right
   - d. Merge the two subarrays into 1 sorted

Disadvantage
- Requires a temporary array



Quicksort $O(n \log n)$
1. If at least 2 elems
   - a. Partition the array
   - b. Quicksort left
   - c. Quicksort right

★ Pivot (usually a[0])
Move markers up & down until vals less than & greater than pivot found. SWAP THEM.

Sequential Search
1. Best case: First slot
2. Worst case: Last slot or not in list
3. Avg: n/2 comparisons

Binary Search
- Has to be SORTED
- When high & low cross, there are no more elements to examine & key is NOT in the arr.
- $\log_2 n$ comparisons & MAX

Robustness - Error Checks
- → No inaccuracy for some input data
- → Will not crash if data is invalid
- → Won't allow execution to proceed if invalid data

Classes - NOUNS
Methods - VERBS

Encapsulation - the process of building a group of methods & fields into a class.
- → SubMethods
- → Assertions (pre & post conditions) ↑ THIS after

Ch 5 - Design & Analysis

Waterfall Model

Analysis of specification → Program Design → Implementation → Testing & Debugging → Maintenance

UML
- diagrams
- unified modeling language

Test Data
1. If pos, test neg
2. Before endpoint, 1
3. At endpoint,
4. After endpoint,

EACH PART OF THE DOMAIN

Efficiency
- Memory
- CPU Time

OOP Design
1. Identify classes to be written
2. Identify methods
3. Determine relationships between classes
4. Write the interface for each class
5. Implement the methods

Inheritance - is-a
Composition - has-a

Classes
- Bottom-Up: Implement lowest level, independent classes first
- Top-Down: Implement main classes first, subsidiary classes later

(Procedural Abstraction - helper methods, stepwise refinement)