Comp 202 Crib Sheet
by Julian Lore Side 1 of 2

Binary first number determines sign, 0001 positive 1, 1111 negative 1 (count backwards)

```
  1 → on      |   0 → off
 001 → 1      |  010 → 2
 011 → 3      |  100 → 4 ...
 101 → 5      |
```

Binary is base 2
Add 1 to last digit, move to next if passes 1

Bases 5123 in Base 10 = $5(10^3) + 1(10^2) + 2(10^1) + 3(10^0)$ | 11001 in Base 2 = $1(2^4) + 2(2^3) + \ldots + 1(2^0)$

1.Structure

Braces {beginning of a block} | Missing/extra {}= compiler er | All { must be in a class/interface

class public class HelloWorld{..}

method Inside classes | public class HelloWorld({public static void $\underbrace{main}_{\text{name}}$ String[] args){}}}}|method inside class → method belongs to that class
Main method: Exec starts at main method, need this to run

Cannot define method inside another

Statements Statements/commands, all end in **;**

What you put in () for commands is **evaluated**

2. Variables
Place in mem reserv to store val. Java: give variables **name** and **type**

Why? Store partial results, generalize code, easier to understand
Names can contain **letters, numbers, and _** |
Names should explain purpose | Convention: degreesInFahrenheit
Can also define your own types of data

Creating var Need 1) Var's type & 2) Var's name → called **declaring** a variable. i.e. int fahrenheitNumber;
Declaring variables important, comp needs to know how much mem to allocate & makes easier for you
Setting value of variable: variable = expression; | i.e. fahrenheit=212; |express will be eval before assignd
Set val of undecl var = compiler error
"String literal". takes as string, if you put var in quotes won't take var

!! Vars made in 1 block **not rel** to vars in other block!

Cmd line args public static void main (String[] args)
args is a var, set by comp when prog starts
Type of args is String[] (String array)
1st String accessed by args[0], 2nd String by args[1] ...
run Test $\underbrace{100}_{args[0]}$ $\underbrace{50}_{args[1]}$
If u parse args that aren't type parsed → runtime error

3. Types
All data manipulated has type, esp. vars

Primitive types These don't ref address!

int : Store integers (32-bit/4 bytes)
byte: Int from -128 to 127 (8-bit)
short: Int -32,768 to 32767 (16-bit)
long : Int from -2^{63} to $2^{63} - 1$ (64-bit)
denoted by L after val (literal), i.e. 3L

double : Store frac/decim, not inf precision, limited decim
To store dbl, comp stores 2 ints, int base $* 2^{\text{int pow}}$
Rounding issues, limited space
# finite in base10 may be inf in base-2 i.e. $\frac{1}{10}$
float: Smaller double (32 bit instead of 64 bit)
denoted by f, i.e. 3f is float
flt f = 3.0 doesnt compile, flt f=3.0f does

boolean : Store true/false
boolean aceExam = true;
Logic ops for bools: &&→ and, ||→ inclusive or, !→ not

char : Store 1 symb (rep by #)
Litral chars denoted by '', i.e 'M' To store ', write '\''

Reference vs primitive types

Primitive: int x=10; → 1) mk space in mem to store int, 2) if we use x, want what is there, 3) store 10 in that mem loc

Reference: int[]x = {1,2} → 1) mk space in mem to store adrs of array, 2)use x, want to acss space in mem (adrs), 3)Mks array elsewhere in mem that stores 1,2 w/ length 2, must be elsewhere, has adrs a, 4) set val of x to a (adrs)
int[] x = {1,2}; int[]y=x; y[0]=2; (x[0]) Will b 2 cuz x & y pt to same, changing inside what y pts to will change for x too

Swapping
Swap prim in other meth, doesnt change in meth that called
Swap ref in another, same thing (cannot change adrs in other meth)

But if you change vals pted @ in othr meth, will change vals pted @ in meth
...swap(int[] a, int[] b){int temp = a[0]; a[0]=b[0]; b[0]=temp} Swaps 1st el of both orig arrays but a=b doesnt change anything in meth

String : store multiple symbols
Lit strings store in " ", " " is empty string
Can combine strings with +
New line w/ "Bla \ n lower bla"
name.length(); for length (diff from arrays, has ())
.charAt(int i);→ i^{th} symb of String (count from 0)
.substring(a,b)→gets String from a up to (not incld) b, also .substring(a)→ a to end
.compareTo(otherStng); → Tell which larger
.indexOf(char c);→ Gives index of 1st c
.toLowerCase();→ mks new string that is lowercase vers (assn to something)
Can def ur own type!
Store type X in thing of type Y, compiler error

4.Expressions Can come in several ways:
1) Literal 10, "hello", true, 3.0
2) Variables someVariable
3) Returned val of non-void methods Math.sqrt(8);
4) Combine expressions

Evaluating expressions
Literals evaluate to type of literal
Type of var is what it was **when you declared it**

Operations
Evaluating multiples expression + with each other, go from less inclsv to more inclusive, widening conv rather than narrowing conv, no data lost
int+int→int, double+double→double, int+double→double, String+String→String, String+int→String
Want narrowing conv, need to cast
int x = (int) 7.5; → x is 7 (truncates)
Casting temp, only changes for expression it is in
double y= 3.5, int x = (int) y, y is still a double
double x = (double)1/2 → 1.0/2 → .5
double x = (double)(1/2)→(double)0→0

Order of Operations Like math
parenthesis→Mult,div,modulo→Addition, subtraction→Assignments

Goes from left to right!
"Your number is"+1+2→ "Your number is12"
"Your number is"+(1+2)→"Your number is3"
1+2+"is your number"→ "3is your number"
int x = 10; ...x=x+1; → x=11
1/2+1/2 → 0 because of trunc
double x = 1/2 → 0, because int divide before assignment
rather: double x = 1.0/2.0 or 1.0/2 or 1/2.0 or 0.5

Other operators -,*,/, % (mod)
Some operators not defined on certain outputs, String*String → Error

INT DIVISION TRUNCATES
9/2 → 4 (decimals cut off! no rounding!)
1/0→ error
Constants: holds on val for entire existence
final double PI = 3.14;
Assign a val to PI again or decl same final again, compiler error

5. Creating new methods
2 types of methods, methods someone else wrote (library methods) or methods you wrote
Writing own allows to group many cmds into 1
Inside a class

Method header is where you give names and stuff for a method like so:
public static **int** **anotherMethod**(**double a**)
int rep output/return type, void if none, var type otherwise. If not void, return blabla; will return to method that called method with value of blabla
return statement has to be reached, only have 1 return in an if, compiler error, because a return has to be able to be reached!
2 ifs, 1 return in each, not compile! 1 if, 1 else, 1 return in each, will compile, cuz return reached no matter what
Only 1 return statement will be reached during an exec of a method, because method is left once it hits 1st return
method name
Input type, name it will rep inside this meth
New meth defs new cmd can use in program
Some just do things: robot.move();
Others give values: double x = Math.sqrt(40);
if meth belongs to same class as meth calling, don't need to write class, as opposed to lib methods
Loc of meth (before or after current meth), does not matter, will scan through whole prog

Remember declaring a var in a meth and trying to use in a diff method → compiler error!
Advantages: code reusability, reduce code dupe, easier debug, problems decomposed, hides tricky logic, easier to read and understand
Disadvantage: a little overhead to set up in beginning (not really)
Modifying given values in another meth will not affect the vals of the meth calling it (unless return modified value and assign it)

6. if
Block of code, only execs if condition is true
if($\underbrace{\text{condition}}_{\text{boolean}}$){ Block, conditional code}
Want something to happen if true, something else if false. Can use 2 if statements (not efficient/clear/good and both or neither can happen). Instead use **if/else**
if(condition){code happens if true}
else{ code happens if false}
Multiple options: if/ else if /else
Instead of nesting ifs inside of elses, use **else if**
if(condition 1){if true} else if(condition 2){if 1 false and 2 true} else if(condition 3){if 1,2 false, 3 true} ...else{all false}

Don't need to end with else for else if
Big difference if we change order of else if
If cond 2 only true if cond 1 true, cond 2 will never be reached
If braces omitted for if statements, compiler assumes braces around 1st statement after condition
Indentation ignored, only for readability
; after if condition → if does nothing i.e. if(x<0); {happens no matter what}

7. boolean
Evaluate to either true or false
boolean expression is either: 1)true or false 2) variable with type boolean 3) call to method that return boolean 4) operator which returns boolean (==,!=,<,>,<=,>=. &&, etc.)

operators
== and != work with any type, inequalities only work with numeric types (or char)
if ops have diff types, lower precision gets promoted, i.e double==int → double==double
and &&, or ||, not ! are boolean operators, take booleans, return booleans
Can combine these together
if(3<x<10) will **not** compile, comp can only do 1 thing at once (3<x)<10→ (tr/fls)<10 → ???error
Fix: if(3<x && x<10)

Comparing chars
Chars have numerical values, unicode number compared
Note: 26 lowercase letters in a row, also 26 uppercase letters all in a row, so 'a'<'c', 'A'<'C'

"short-circuit "evaluation
Left of && false, remaining operands not looked at
If left of || true, remaining not looked at
Good for speed
if(bool==true) useless, if already checks if true, so: if(boolean)

Remember x=5 is **not** a boolean! That's an assignment, need ==

Comparing floating pts
Shouldn't use equality operator for float/double
Might think they're close enough, but not exactly equal
.1+.1+.1==.3 → false
Alternatively: get abs val of difference between 2 doubles, say they must be less than some small decimal

Comparing reference types
To compare two ref types, do not use == or !=, as that will compare address! Use a.equals(b);
If you use a==b, you'll get false, if adrs are diff (if both point to 1,2,3, but in diff parts of memory)
i.e. int[] x={1,2}; int[]y=x; x==y→ true but int[] x={1,2}; int[] y={1,2}; x==y→ false

8. Loops

while Execute forever while condition is true
while(cond){Keeps happening until false}
Code after only happens once loop done, cond false
Loop counter: int i=0; while(i<5){Do something; i++;} Will do this 5 times
Cond only eval at begin of loop, not after every statement, i.e. can add 10 to i and then subtract 10 from i before loop ends, will still go on
Infinite loops=rip
while(cond); is an infinite loop! Not executing anything

for Common loop theme has 3 steps, initialization, iteration (of loop) given condition, finalization
int x = 0; (initialize) while(x<4)(condition){Loop action; x++; (finalization at end)}
for(initialization;condition; finalization){loop body}
initialization happens once, before first step of loop | condition same as while loop | finalization last step of loop (must be a valid statement! i=i+2 instead of i+2)
Copy of while loop: for(int x=0; x<4; x++){loop action;}
Benefits: more readable

nested loops Loop inside of another loop

9. Arrays
Array container object, holds fixed # of vals of 1 type, length fixed & established at creation
Many vals of same type into 1 array (1 "object ")
Creating: type[] arrayName = new type[int of size]
Reserves mem, n places for size n, all in a row, all assigned an index or explicitly declare array entries: type[] arrayName = {1,2,3}

String[] args in main method is an array!
Access entries: arrayName[0], arrayName[1],...
Set values of entries: arrayName[0]=1;
arrayName.length → int giving # of elements

Arrays of Arrays
arrayName[array number][number within array] i.e. a[1][5] gives 6th element of 2nd Array
multidimArray.length → # of arrays in array
Want # of elem in an array, multidimArray[index of array wanted].length

Comp 202 Crib Sheet
by Julian Lore Side 2 of 2

Declare multidim array: int[][] arr = {{1,2,3},{1,4}};
or: type[][] name = new type[size 1][size 2]; (makes rectangular array, all same size)
Can also make jagged arrays: type[][] name = new type[5][]; Can put diff size arrays in this
Can have more arrays in arrays type[][][][][][]...

Printing an Array

System.out.println(array); prints address in mem

Packages

Can import libraries from packages! Need import java.util.Arrays; in preamble, before class
Packages contain 1+ classes, class contain 1+ methods, methods have 1+ commands
Look up class, gives you package name. If package ≠ java.lang, need to import
Arrays.toString(int[] x) → {1,2} or whatever its contents are, then print this
Null: literal value that can be used for all ref types → points to nowhere
int[] a; by itself does not work, undeclared array. But int[] a=null; does
if a null, a[0] or a.length→RuntimeError (NullPointerException, occurs whenever you use . or [] on null)
10. Exceptions
Impossible to exec → exception or runtime error
Info from exceptions:
Exception in thread "main" java.util.InputMismatchException
at java.util.Scanner.throwFor (Scanner.java:819)
Type of exception
Stack trace, which methods called method that crashed prog
Line number and file that exception occured in

Array exceptions

ArrayIndexOutOfBounds → try to access invalid index, exception gives index number that caused it
NullPointerException (access properties of null var)

Throwing Some commands can't be executed given certain input, instead of using ifs and whatnot to make output null, should throw, problem will become hard to find if not
Immediately generate an error by using throw, rather than hide
if(stuff==null || ...){throw new IllegalArgumentException("Invalid input ")}
Other kinds of exceptions: DivideByZero, NumberFormatException (String to number, etc.)
With throw, method will give value or error
If you want to process the job, use try/catch
try some stuff; catch(IllegalArgumentException e){Happens if that type of exception happens, if no error, skip this, if different type of exception, pass to caller}
Diff types of Exceptions, hierarchy
Can use catch(Exception e){ To catch all exceptions} | Dangerous though! Can hide all bugs

Can put multiple catch blocks after, first one that matches will be executed
Each method in method chain can catch exception: main()→a()→b()→c()
If c gens exception, can try/catch in b. If not, can try/catch in a, if not, then main. If none catch, then prog crashes
Can also use try/catch/finally, finally will occur no matter what, good for: removing dupe code, clean up before crash/return
Can even do try/finally

3 types of errors Compile time, run time, bug(incld infinite loop) → compile gives most info, run a bit, bug none
Throw used to give runtime error rather than bug
Exception is an object, catch(Exception e) declares var e of type Exception
Can create one via new Exception("info "), make own exception type for particular kind of prob in your code
11. Scanner
Need to import library, import java.util.Scanner;
Object, ref type (like array & exception)
Declare new: Scanner reader = new Scanner(System.in); System.in to get input from user
Read vals from user: scannerName.nextInt(); .nextDouble(); .next(); (until space) .nextBoolean();

Reading from file

Scanner fileRdr = new Scanner(new File("foo.txt")); foo in same folder as .class file | need to import java.io
Need to deal with exception from this! Need to import java.io.IOException;
try{... new File..} catch(IOException e){..}
Can just import java.io.*

Mandatory Exception Handling

FileNoteFoundException is a "checked exception", mandatory to catch or add to method header: public Scanner(File f) throws FileNotFoundException{}
Unchecked exceptions are the usual, like ArrayIndexOutOfBoundsException, etc.
To write to a file, use FileWriter and BufferedWriter: (Buffered to write temp to RAM, faster but not necessary)
Put this inside try/catch or throw in header
FileWriter fw= new FileWriter("file.txt"); if file.txt exists, will overwrite. If not, creates
BufferedWriter bw = new BufferedWriter(fw);
bw.write(StringName); bw.newLine();
bw.write(3+" ");
bw.close(); close it, will get written to fw
fw.close(); always close at end
12. User defined types

```
public class NameOfClass{
    modifier1 type1 name1;
    modifier2 type2 name2
    ... }
```

private vs public

public meth or prop → any meth or prop can be accessed directly from any class
private → meth or prop only accessed directly from class defined in
try to access from a diff class → compiler err
Private properties gives more flexibility over time
Should usually have private attributes

```
public class Cat {
    private String name;
    private int age;
    private boolean isGiantCat; }
```

Can make 1 prop an array, other props an array entry, make things easier to loop through
Use getters and setter methods
public String getName(){return this.name;} | public void setName(String newName){this.name=newName;}
To use new type, create a 2nd class in same folder, can mk var of new type inside that class
Ref types, use new operator
Cat garfield = new Cat(); garfield refs to obj w/ 3 vals
If attributes were public, could directly modify garfield.name="Garfield";
Static vs not static: Math.PI; → Math is name of class, static | String s="foo"; s.length() → length non-static, needs name of string
Attributes for new obj non static
Can make new methods for object: public void talk(){..} → garfield.talk;
Inside non static meth, keyword this access obj meth called on
public void talk() {if (this.isGiantCat){Stuff} else{..}} (can also omit this in a non-static but bad syntax)
But if omitted this makes it the same as another var, will not conv to this.varname!
Cannot call this in a static meth

constructor

By default, sets all props null, can define your own
No return type, must have same name as class
public Cat()this.age=10; | public Cat(int initialAge){this.age=initialAge;}
Default constructor is gone once you make your own
Can also make a private constructor, like Math, if don't want anyone to make one
Writing own constructor can help avoid bugs, like getting a prop before you set it

mutable vs immutable

mutable object → obj can change | array mutable, can change entries | string immutable, cannot change certain chars, can change string, but makes a new string at diff adrs
If return an address of mutable (ex. Array), can be changed in other meth!
Unnecessary setters: setting a prob in your constructor and having a setter? Bad → 2-1 alias issue
If method has nothing to change contents of existing obj (only has constructor), don't need to worry about aliasing → like String
Can chain props d1.times.length;
Example of a new type

DynamicArray

Want to create a new object that's an array whose size doesn't need to be specified
Have String[] array and int numElements prop | Constructor gives array fixed size, say 100, numEl 0 | size method to return numEl | get(int i) to return data at that point | add(String s) to assign numElements to s and ++numEl, use an if to check if numEl==length of data, if it is, make bigger array and recopy everything back | Should also simulate array errors, like ArrayIndexOutOfBounds | equals(DynamicArray array) | remove(index) | remove(String) | indexOf(String) | contains(String)

Use Generics to generalize to any type
Every new obj extends obj, has pre-defined methods for obj (Like println, toString())
By default, contains uses ==
Can override these methods by making new one
13. Generics
Finding first String s in a String[] requires .equals(), but first int x in a int[] requires == (also, can't write a method for both, as 2 diff types of array and 2 diff types searching for)
Can add 1 or more generic type specs in defining class
public class DynamicArray<T> {private T[] data; public T get(int i){..}}
T can be replaced by any class, creating an obj of gen type, specify class
DynamicArray<String> foo = new DynamicArray<String>();
But, cannot use with primitive types!
Sol'n: Wrapper class: make a new type, ex. IntReference that consists of just the prim type, but now refers to that
Integer is a default wrapper in java.lang (no import)
Integer fiveAsInteger = new Integer(5); | int fiveAsInt=fiveAsInteger.intValue();
As of Java 1.5, automatic boxing and unboxing, can use int and Integer interchangeably | Integer i = 3;
Strings immutable, constantly adding (summing) to String has to regenerate the String each time, slow
Adding 1000 stars: 500 pairs, (1+1000), (2+999), .., 1001*500=500500 $\frac{n(n+1)}{2}$ with $n = 1000$ amount of
Alternatively, char array of 1000, only 1000 steps to assign 1000 stars
Can then make a new type StringBuilder, like DynamicArray but for char arrays | can use toString() to conv char[] | Java already has StringBuilder

ArrayList

Need to import java.utils.Array;
Java's vers of DynamicArray, resizable array for any type
new ArrayList<obj_type>(); | ex. ArrayList<String> words=new ArrayList<String>(); words.add("Word"); words.get(0);
E→ type | add(E e); | add(int index, E elm); | clear(); removes all ele from list | get(int index); | indexOf(obj O); | remove(int index);| remove(Obj o); | size(); | contains(obj);
ArrayList is ref type, stores links to obj, using .get(); gives you another link, potential aliasing prob
Getting a ref val of ArrayList, then calling a method that changes prop of that obj will change it for the ArrayList too!
Don't need to worry about this with ArrayList of String, immutable

Commands

Printing

System.out.println(); Prints on new line
System.out.print(); Prints

Parsing

Integer.parseInt(String a); String→ int
Double.parseDouble(String a); String→ double

Math library

Math.sqrt(double a); Returns square root of number
Math.PI; Approximation of Pi (double)
Math.abs(double a); Returns absolute value of a number
Math.pow(double a, double b); Returns a^b
Math.sin(double a); Returns sin of a (rads)
Math.random(); Returns random double $0.0 \leq double < 1.0$

Misc

++x; or x++→(x=x+1);
–x; or x–→(x=x-1);
x(op,i.e. +)=9 → (x=x(op)9)
Integer.MAX_VALUE; gives integer max value
Integer.MIN_VALUE; gives integer min value

Examples

What's wrong with this?

```
int x = Integer.parseInt(args[0]);
if(x<0){
    System.out.println("Positive #")
}
if (x>0){
    System.out.println("Negative #")
}
else{
    System.out.println("0 ")
}
```

If the first if is true and the second false, the else will still print! Else only affects the if above. To fix, change second if to else if
Can't initialize array 2x (even if 1 declares size, the other declares entries)
Misc Var with same name can be declared 2x in same method (i.e. 2 for loops)
Index out of bounds → run time error , printing in a non-void method will still print
You can make two arrays of diff size equal to another, just changes address it points at, int[] a= new int[5]; int[] b = new int[10]; a=b; will work!
Remember negation distributes! !(a || b)→!a&&!b
Bytecode is result of compiling
Constant doesn't need memory
a && b and b&& a don't eval to same result, if first is false, says error, maybe first gives compiler error instead
Compiler error → everything looks fine, runtime error → logic doesn't work
Can compile java file without main method
Default val of int[] is 0 in each pos
System.out.println(..); is a non static method (called on out)
Can overload methods, make 2 methods with same name, but diff input, will call suitable one
Cannot overload methods with diff return type, compiler error
Every java prog has at least 1 var declared
A non-static method can access a static var in same class
ArrayList cannot store *different* types of values
Can assign a line from reader to String[], can also use name.split(String) to split around matches of that String
"\t" is a tab character for String
Trying to access uninitiated String[] args is an ArrayIndexOutOfBoundsException
Out of bounds is a runtime error → will still compile!
Writing a diff method that is static vs non-static is not considered overloading
Can initialize double with Double d = new Double(29.95);