#_ Important [Java Built-in Methods] {CheatSheet}

1. String Methods

- **length**: `str.length()`
- **substring**: `str.substring(beginIndex, endIndex)`
- **charAt**: `str.charAt(index)`
- **contains**: `str.contains(substring)`
- **indexOf**: `str.indexOf(substring)`
- **toLowerCase**: `str.toLowerCase()`
- **toUpperCase**: `str.toUpperCase()`
- **trim**: `str.trim()`
- **replace**: `str.replace(oldChar, newChar)`
- **split**: `str.split(regex)`
- **startsWith**: `str.startsWith(prefix)`
- **endsWith**: `str.endsWith(suffix)`
- **equals**: `str.equals(anotherString)`
- **equalsIgnoreCase**: `str.equalsIgnoreCase(anotherString)`
- **compareTo**: `str.compareTo(anotherString)`
- **isEmpty**: `str.isEmpty()`
- **concat**: `str.concat(anotherString)`
- **valueOf (static)**: `String.valueOf(data)`

2. Math Methods

- **abs**: `Math.abs(x)`
- **max**: `Math.max(x, y)`
- **min**: `Math.min(x, y)`
- **sqrt**: `Math.sqrt(x)`
- **pow**: `Math.pow(x, y)`
- **round**: `Math.round(x)`
- **random (static)**: `Math.random()`
- **ceil**: `Math.ceil(x)`
- **floor**: `Math.floor(x)`
- **sin**: `Math.sin(angle)`
- **cos**: `Math.cos(angle)`

By: Waleed Mousa

- **tan**: Math.tan(angle)

3. System Methods

- **currentTimeMillis (static)**: System.currentTimeMillis()
- **nanoTime (static)**: System.nanoTime()
- **exit (static)**: System.exit(status)
- **gc (static)**: System.gc()
- **getenv (static)**: System.getenv(variableName)
- **getProperty (static)**: System.getProperty(propertyName)

4. Object Methods

- **equals**: obj.equals(anotherObj)
- **hashCode**: obj.hashCode()
- **getClass**: obj.getClass()
- **toString**: obj.toString()
- **clone (protected)**: obj.clone()
- **notify**: obj.notify()
- **notifyAll**: obj.notifyAll()
- **wait (overloaded)**: obj.wait()

5. Collection Methods (List, Set, Map)

- **add**: collection.add(element)
- **remove**: collection.remove(element)
- **contains**: collection.contains(element)
- **size**: collection.size()
- **isEmpty**: collection.isEmpty()
- **iterator**: collection.iterator()
- **clear**: collection.clear()
- **addAll**: collection.addAll(anotherCollection)
- **removeAll**: collection.removeAll(anotherCollection)
- **retainAll**: collection.retainAll(anotherCollection)
- **toArray**: collection.toArray()

By: Waleed Mousa

6. Map Methods

- **put**: `map.put(key, value)`
- **get**: `map.get(key)`
- **remove (overloaded)**: `map.remove(key)`
- **containsKey**: `map.containsKey(key)`
- **containsValue**: `map.containsValue(value)`
- **keySet**: `map.keySet()`
- **entrySet**: `map.entrySet()`
- **values**: `map.values()`
- **size (overloaded)**: `map.size()`
- **isEmpty (overloaded)**: `map.isEmpty()`

7. Thread Methods

- **start**: `thread.start()`
- **run**: `thread.run()`
- **join**: `thread.join()`
- **sleep (static)**: `Thread.sleep(millis)`
- **currentThread (static)**: `Thread.currentThread()`
- **setName**: `thread.setName(name)`
- **getName**: `thread.getName()`
- **setPriority**: `thread.setPriority(priority)`
- **getPriority**: `thread.getPriority()`
- **interrupt**: `thread.interrupt()`
- **isInterrupted**: `thread.isInterrupted()`

8. IO and NIO Methods

- **read (FileInputStream, BufferedReader, etc.)**: `stream.read()`
- **write (FileOutputStream, BufferedWriter, etc.)**: `stream.write(byte[] b)`
- **close (AutoCloseable interfaces)**: `resource.close()`
- **flush (OutputStream, Writer)**: `writer.flush()`
- **open (Files in NIO)**: `Files.open(path, options)`
- **copy (Files in NIO)**: `Files.copy(source, target, options)`
- **delete (Files in NIO)**: `Files.delete(path)`

By: Waleed Mousa

- **create (Files in NIO)**: `Files.createFile(path)`
- **readAllBytes (Files in NIO)**: `Files.readAllBytes(path)`
- **write (Files in NIO, overloaded)**: `Files.write(path, bytes, options)`

9. Networking (java.net package)

- **connect (Socket)**: `socket.connect(address)`
- **send (DatagramSocket)**: `socket.send(packet)`
- **receive (DatagramSocket)**: `socket.receive(packet)`
- **bind (ServerSocket)**: `serverSocket.bind(endpoint)`
- **accept (ServerSocket)**: `serverSocket.accept()`
- **getInputStream (Socket)**: `socket.getInputStream()`
- **getOutputStream (Socket)**: `socket.getOutputStream()`
- **lookup (InetAddress, static)**: `InetAddress.getByName(host)`
- **getHostAddress (InetAddress)**: `inetAddress.getHostAddress()`
- **getHostName (InetAddress)**: `inetAddress.getHostName()`
- **isReachable (InetAddress)**: `inetAddress.isReachable(timeout)`

10. Date and Time (java.time package)

- **now (LocalDate, LocalDateTime, Instant)**: `LocalDate.now()`
- **of (LocalDate, LocalDateTime)**: `LocalDateTime.of(year, month, day, hour, minute)`
- **parse (LocalDate, LocalDateTime)**: `LocalDate.parse("2020-01-01")`
- **format (DateTimeFormatter)**: `dateTime.format(formatter)`
- **plusDays, plusHours, etc. (LocalDate, LocalDateTime)**: `dateTime.plusDays(1)`
- **between (Duration, Period)**: `Duration.between(startTime, endTime)`

11. Exception Handling

- **printStackTrace**: `exception.printStackTrace()`
- **getMessage**: `exception.getMessage()`
- **getCause**: `exception.getCause()`

12. File Handling (java.io package)

By: Waleed Mousa

- **createNewFile (File)**: `file.createNewFile()`
- **mkdir (File)**: `file.mkdir()`
- **listFiles (File)**: `file.listFiles()`
- **exists (File)**: `file.exists()`
- **canRead (File)**: `file.canRead()`
- **canWrite (File)**: `file.canWrite()`
- **length (File)**: `file.length()`
- **delete (File)**: `file.delete()`

13. Advanced Collection Operations

- **stream (Collection)**: `collection.stream()`
- **forEach (Iterable)**: `collection.forEach(item -> { /* action */ })`
- **removeIf (Collection)**: `collection.removeIf(item -> item.condition())`
- **sort (List)**: `list.sort(Comparator.naturalOrder())`
- **collect (Stream)**: `stream.collect(Collectors.toList())`
- **map (Stream)**: `stream.map(item -> item.transform())`
- **filter (Stream)**: `stream.filter(item -> item.condition())`
- **findAny, findFirst (Stream)**: `stream.findAny()`
- **reduce (Stream)**: `stream.reduce((acc, item) -> acc.combine(item))`

14. Concurrency and Multithreading

- **start (Thread)**: `thread.start()`
- **run (Runnable)**: `runnable.run()`
- **submit (ExecutorService)**: `executorService.submit(callable)`
- **shutdown (ExecutorService)**: `executorService.shutdown()`
- **awaitTermination (ExecutorService)**: `executorService.awaitTermination(time, unit)`
- **isShutdown (ExecutorService)**: `executorService.isShutdown()`
- **synchronized methods or blocks**: `synchronized(obj) { /* synchronized code */ }`

15. Serialization

- **writeObject (ObjectOutputStream)**: `out.writeObject(object)`

By: Waleed Mousa

- **readObject (ObjectInputStream)**: `Object obj = in.readObject()`

16. Reflection

- **getMethod (Class)**: `cls.getMethod(methodName, paramTypes)`
- **newInstance (Class)**: `cls.newInstance()`
- **getDeclaredFields (Class)**: `cls.getDeclaredFields()`
- **invoke (Method)**: `method.invoke(obj, args)`

18. Regular Expressions (java.util.regex package)

- **matches (Pattern and Matcher)**: `Pattern.matches(regex, input)`
- **find (Matcher)**: `matcher.find()`
- **group (Matcher)**: `matcher.group()`

19. Java NIO (java.nio package)

- **allocate (ByteBuffer)**: `ByteBuffer.allocate(size)`
- **put (ByteBuffer)**: `byteBuffer.put(data)`
- **get (ByteBuffer)**: `byteBuffer.get()`
- **flip (Buffer)**: `buffer.flip()`
- **clear (Buffer)**: `buffer.clear()`

20. Java Utility Methods

- **random (Random)**: `new Random().nextInt()`
- **shuffle (Collections)**: `Collections.shuffle(list)`
- **copy (Collections)**: `Collections.copy(dest, src)`
- **fill (Collections)**: `Collections.fill(list, obj)`
- **singletonList, singletonMap (Collections)**: `Collections.singletonList(item)`
- **unmodifiableCollection (Collections)**: `Collections.unmodifiableCollection(collection)`

By: Waleed Mousa