Cheatography

Java Fundies 2 Cheat Sheet

by Edgar (edgarjoel21) via cheatography.com/44289/cs/13117/

Contains (CartPt)

```
public boolean contains(CartPt pt) { return (this.nw.x <= pt.x) && (pt.x <= this.nw.x + this.size) && (this.nw.y <= pt.y) && (pt.y <= this.nw.y + this.size); } }
```

This occurs when we have CartPt nw as a field

String Methods

s.length()	s.charAt(/)
s.substring(start, end)	s.toUpper-Case()
s.toLowerCase()	s.indexOf(x)
s.replace(old, new)	s.split(regex)
s.trim()	s.equals(s2)
s.compareTo(s2) (Compares two strings Lexiographically)	s.contains(CharSequence cs)

Arithmetic Operators

x + y	add	x - y	subtract
x * y	Multiply	x/y	divide
x % y	modulo		

Boolean Operators

! x	not
x && y	and
x || y	or

Abstracted Sameness

```
abstract class AFoo implements IFoo { public boolean sameX(X that) { return false; } public boolean sameY(Y that) { return false; } public boolean sameZ(Z that) { return false; } } class X extends AFoo { public boolean sameFoo(IFoo that) { return that.sameX(this); } public boolean sameX(X that) { ... compares two X values ... } } class Y extends AFoo { public boolean sameFoo(IFoo that) { return that.sameY(this); } public boolean sameY(Y that) { ... compares two Y values ... } } class Z extends AFoo { public boolean sameFoo(IFoo that) { return that.sameZ(this); } public boolean sameZ(Z that) { ... compares two Z values ... } }
```

Comparisons

x < y	less	x <= y	less than or equal
x > y	greater than	x >= y	greater than or equal

Comparisons (cont)

x == y	Equal	! x	Not equal

Distance formula (Origin)

```
public double distanceToOrigin() { return Math.sqrt(this.x *this.x + this.y* this.y) - this.radius; } }
```

- this.radius is only for circle

DistanceTo Formula

```
class CartPt { ... double distanceTo(CartPt that) { return Math.sqrt( (this.x - that.x) *(this.x - that.x)* + *(this.y - that.y)* (this.y - that.y)); } }
```

Insert()

```
public ILoBook insert(Book b) { if (this.first.cheaperThan(b)) { return new ConsLoBook(this.first, this.rest.insert(b)); } else { return new ConsLoBook(b, this); } }
```

ConsLoBook()

```
public ILoBook insert(Book b) { return new ConsLoBook(b, this); }
```

MtLoBook()

append(ILoItem other)

```
public ILoItem allNumbers() { return this.left.allNumbers().append(this.right.allNumbers()); }
```

```
// In interface -- ILoItem append(ILoItem other);
```

```
// in consLoItem -- public ILoItem append(ILoItem other) { new ConsLoItem(this.first, this.rest.append(other)); }
```

```
// MtLoItem --- public ILoItem append(ILoItem other) { return other; }
```

Younger IAT (In IAT)

In IAT

```
IAT youngerIAT(IAT other); ------- IAT youngerIATHelp(IAT other, int otherYob);
```

Younger IAT and Helper

```
public IAT youngerIAT(IAT other) {return other.youngerIATHelp(this, this.yob);
```

Younger IAT and Helper (cont)

```
IAT youngerIATHelp(IAT other, int otherYob) { if (this.yob > otherYob) { return this; } else { return other; } }
```

Youngest Parent

```
public IAT youngestParent() { return this.mom.youngerIAT(this.dad); }
```

Youngest GrandParent

```
public IAT youngestGrandparent() { return this.mom.youngestParent().youngerIAT(this.dad.youngestParent()); }
```

Abstract Interface

```
abstract class AShape implements IShape { CartPt loc; String color;
  AShape(CartPt loc, String color) { this.loc = loc; this.color = color; } }
```

Abstract Class

```
class Circle extends AShape { int radius;
Circle(CartPt center, int radius, String color) { super(center, color); this.radius = radius; } }
```

Subclass entension

```
class Square extends Rect {   Square(CartPt nw, int size, String color) { super(nw, size, size, color); }
```

size, size represent length and width.
You would need to override the method to use size rather than length/width

Tips

Don't:	Do:
Casting	Design Recipe
Field of Field	Tests: test helpers
isFoo() (isEmpty())	Purpose statements
Getters	Shorthand
	Dynamic Dispatch

Abstract with Range in constructor

```
interface ITetrisPiece { int SCREEN_HEIGHT = 30; }

abstract class ATetrisPiece implements ITetrisPiece { ... ATetrisPiece(int x, int y) { this.xPos = x; this.yPos = y; } ATetrisPiece(int x) { this(x, SCREEN_HEIGHT); } }
```

Utilis Class

```
class Utils { int checkRange(int val, int min, int max, String msg) { if (val >= min && val <= max) { return val; } else { throw new IllegalArgumentException(msg); } } }
```

This allows a constructor to be general

Illegal Exception Message in Constructor

```
// In class Date ----------Date(int year, int month, int day) { this.year = new Utils().checkRange(year, 1500, 2100, "Invalid year: " + Integer.toString(year)); this.month = new Utils().checkRange(month, 1, 12, "Invalid month " + Integer.toString(month)); this.day = new Utils().checkRange(day, 1, 31, "Invalid day: " + Integer.toString(day));
```

integer.toString(year))
changes the invalid integer to a string and combines (+)

Testing Exceptions

```
// In Tester ---- boolean checkConstructorException(Exception e, String className, ... constr args ...);
```

Sameness Interface

```
interface IShape { boolean sameShape(IShape that); boolean sameCircle(Circle that); boolean sameRect(Rect that); boolean sameSquare(Square that); }

// In Circle: public boolean sameShape(IShape that) { return that.sameCircle(this); }
```

Others would be false in this class