Pickering Lab Class Map

	Class MetaGenome	KEY
	"Each instance takes a .FASTA and turns it into a set of Sequences"	*: Throws errors
P #	def __init__ (metaGenome, name) "Instantiates, given MG .FASTA file"	✓ #: Validates input
	def getName()	✓ P: Private
#	def findOffTargets(spacerSequence) "Uses Sequences FOT method, returns	✓
*	def addSequences(sequences) "Adds sequences from .FASTA"	✓
#	def getSequenceAtIndex(index) def getSequenceWithSubseq(subseq)	✓ ✓
P	Sequence[] __Sequences "List of sequences in metagenome"	✓
P	String __MetaGenomeName "Name of MetaGenome"	✓

	Class Sequence
	"Each instance stores a sequence & info & can analyze it dynamically"
P#	def __init__ (sequence, name) "Takes SeqRecord & String & turns into info"
#	def setMismatchStrictness (mismatches) "Setter function for mismatch strictness"
#	def findOffTargets(spacerSequence) "Returns String[] of off-targets"
#	def setName(name) "Sets String name"
#	def correlateStrings (String1, String2) "Returns list of corr. values, indexed by corr. shifts"
P	SeqRecord __Record "Record class with sequence into"
P	String __SequenceName "Name of sequence"
P	int __MismatchStrictness "# of mismatches allowed in an off-target seq"

	Class sgRNASpacer
	"Each instance takes a spacer sequence & analyzes it, then stores."
P* #	def __init__ (spacerSequence, metaGenome) "Takes in sequence & a metaGenome
P	def __calcHeuristic() "Calculates heuristic from on/off-target scores
	def getHeuristic() "Returns resulting heuristic"
P	def __calcOnTargetScore() "Calculates on-target score"
P	def __calcOffTargetScore (offTargetSequence) "Given off-target, calc oTS"
#	def validDNA(DNAseq) # def validRNA(RNAseq)
P	String __SpacerSequence P double __onTargetScore P double __heuristic
P	String[] __OffTargetSeqs P double[] __offTargetScores