COMPOSITE AND STRATEGY DESIGN FOR HW8 <<Interface>> Comparable SearchEngineRunner e: Engine - listings: List<IListing> s: Scanner <<Interface>> IListing + mainRunner() ListingComparison <<Interface>> + radiusRunner() ListingComparator + RANK: Map + suggestionRunner() - ListingComparator: ListingComparator + LAT: double + getComparator(): Comparator + LON: double + compare(IListing, IListing): int + sort(List<IListing>, ListingComparator.getComparator() + byDescendingOrder(): Comparator<IListing> child + byRadiusDistance(): Comparator<IListing> <<Interface>> + compareTo(Term): int Graph + toString(): String <<Interface>> **IEngine** DescendingOrder Lexicographic DistanceOrder + MAX NUM DISPLAY: int getComparator(): Comparator + getComparator(): Comparator + getComparator(): Comparator Listina - id: int - propertyType: String GraphL - roomType: String listingName: String - nodeArray: Edge[] parent Edge - price: double nodeValues: Object[] Engine - numEdge: int - accommodates: int + vertex: int - lat: double maxAccommodates: int + weight: int - Ion: double maxPrice: double + prev: Edge - numReviews: int + init() maxNumReviews: int + next: Edge - clique: Collection<lListing> + nodeCount(): int epicenter: Listing - score: double + edgeCount(): int - distance: double + getValue(): Object - priceCheck: int + setValue() - distanceCheck: int - find(): Edge + addEdge() - propertyTypeCheck: int - roomTypeCheck: int + getListings(): ArrayList<IListing> + weight(): int - reviewsCheck: int + removeEdge() + outputListings(): Collection<IListing> - accommodatesCheck: int + hasEdge(): boolean + printListings() - radiusDist: double + getPropertyType(): Collection<String> + neighbors(): int[] + getRoomType(): Collection<String> + computeDistance(): double + checkPrice() - getNeighbors(): int[] + checkDistance() + makeGraph(): Graph + checkReviews() + makeClique(): ArrayList<IListing> + checkPropertyType() + userRank(): Map<String, Integer> + checkRoomType() + checkAccommodates()

+ computeScore(): double