Factory method	Returned type	Used to
toList	**List<T>**	**Gather all the stream’s items in a List.**
Example use: List<Dish> dishes = menuStream.collect(toList());		
toSet	**Set<T>**	**Gather all the stream’s items in a Set, eliminating duplicates.**
Example use: Set<Dish> dishes = menuStream.collect(toSet());		
toCollection	**Collection<T>**	**Gather all the stream’s items in the collection created by the provided supplier.**
Example use: Collection<Dish> dishes = menuStream.collect(toCollection(), ArrayList::new);		
counting	**Long**	**Count the number of items in the stream.**
Example use: long howManyDishes = menuStream.collect(counting());		
summingInt	**Integer**	**Sum the values of an Integer property of the items in the stream.**
Example use: int totalCalories = menuStream.collect(summingInt(Dish::getCalories));		
averagingInt	**Double**	**Calculate the average value of an Integer property of the items in the stream.**
Example use: double avgCalories = menuStream.collect(averagingInt(Dish::getCalories));		
summarizingInt	**IntSummary-Statistics**	**Collect statistics regarding an Integer property of the items in the stream, such as the maximum, minimum, total, and average.**
Example use: IntSummaryStatistics menuStatistics = menuStream.collect(summarizingInt(Dish::getCalories));		
joining	**String**	**Concatenate the strings resulting from the invocation of the toString method on each item of the stream.**
Example use: String shortMenu = menuStream.map(Dish::getName).collect(joining(", "));		
maxBy	**Optional<T>**	**An Optional wrapping the maximal element in this stream according to the given comparator or Optional.empty() if the stream is empty.**
Example use: Optional<Dish> fattest = menuStream.collect(maxBy(comparingInt(Dish::getCalories)));		
minBy	**Optional<T>**	**An Optional wrapping the minimal element in this stream according to the given comparator or Optional.empty() if the stream is empty.**
Example use: Optional<Dish> lightest = menuStream.collect(minBy(comparingInt(Dish::getCalories)));		
reducing	**The type produced by the reduction operation**	**Reduce the stream to a single value starting from an initial value used as accumulator and iteratively combining it with each item of the stream using a BinaryOperator.**
Example use: int totalCalories = menuStream.collect(reducing(0, Dish::getCalories, Integer::sum));		
collectingAndThen	**The type returned by the transforming function**	**Wrap another collector and apply a transformation function to its result.**
Example use: int howManyDishes = menuStream.collect(collectingAndThen(toList(), List::size));		
groupingBy	**Map<K, List<T>>**	**Group the items in the stream based on the value of one of their properties and use those values as keys in the resulting Map.**
Example use: Map<Dish.Type, List<Dish>> dishesByType = menuStream.collect(groupingBy(Dish::getType));		
partitioningBy	**Map<Boolean, List<T>>**	**Partition the items in the stream based on the result of the application of a predicate to each of them.**
Example use: Map<Boolean, List<Dish>> vegetarianDishes = menuStream.collect(partitioningBy(Dish::isVegetarian));		