List<String> list = people.stream().map(Person::getName).collect(Collectors.toList());

// Accumulate names into a TreeSet

Set<String> set = people.stream().map(Person::getName).collect(Collectors.toCollection(TreeSet::new));

// Convert elements to strings and concatenate them, separated by commas

String joined = things.stream()

.map(Object::toString)

.collect(Collectors.**joining**(", "));

// Compute sum of salaries of employee

int total = employees.stream()

.collect(Collectors.**summingInt**(Employee::getSalary)));

// Group employees by department

Map<Department, List<Employee>> byDept

= employees.stream()

.collect(Collectors.**groupingBy**(Employee::getDepartment));

// Compute sum of salaries by department

Map<Department, Integer> totalByDept

= employees.stream()

.collect(Collectors.groupingBy(Employee::getDepartment,

Collectors.**summingInt**(Employee::getSalary)));

// Partition students into passing and failing

Map<Boolean, List<Student>> passingFailing =

students.stream()

.collect(Collectors.**partitioningBy**(s -> s.getGrade() >= PASS\_THRESHOLD));

Comparator<String> cmp = Comparator.comparingInt(String::length) .thenComparing(String.CASE\_INSENSITIVE\_ORDER);

Comparator.comparing( Person::getLastName, String.CASE\_INSENSITIVE\_ORDER);