Section 7 - Collections, Streams, and Filters
Quiz Answers

1.

The builder pattern is useful when you have many parameters, some of which are optional. This way, you don't need multiple overloaded constructors.

2.

```
ArrayList<Integer> items = new ArrayList<Integer>();
for (int i = 0; i < 100; i++)
{
      items.add((int) (Math.random() * 100));
}

items.stream().forEach((p) -> System.out.println(p));
```

3.

```
ArrayList<Integer> items = new ArrayList<Integer>();
for (int i = 0; i < 100; i++)
{
      items.add((int) (Math.random() * 100));
}

items.stream().map((p) ->
```

```
{
      int randomCharIndex = (int) (Math.random() * 26);
      char randomChar = (char) (randomCharIndex + 'A');
      return p + Character.toString(randomChar);
})
.forEach((p) -> System.out.println(p));
```

4.

```
ArrayList<Integer> items = new ArrayList<Integer>();
for (int i = 0; i < 100; i++)
{
      items.add((int) (Math.random() * 100));
}

items.stream().map((p) ->
{
      int randomCharIndex = (int) (Math.random() * 26);
      char randomChar = (char) (randomCharIndex + 'A');
      return p + Character.toString(randomChar);
})
.filter((p) -> p.contains("A"))
.forEach((p) -> System.out.println(p));
```