1. The **`Main`** app creates a **`Generator`** object and invokes its **`run`** method.

```
public class Main {
  public static void main(String[] args) {









  }
}
```

2. Whenever the **`Generator`** produces an integer, we want to process it. For example, we can print $*$. We want to *decouple* the processing of the integers from the production of the integers so that we need not make any changes to the **`Generator`** class if we want to change the processing of the integers. Hence, we create a **`StarPrinter`** class with a method **`process`** to print $*$.

```
public class StarPrinter {









}
```

3. Whenever the **`Generator`** produces an integer, it should invoke the **`process`** method on a **`StarPrinter`** object.

```
public class Generator {
  public void run() {
    ...
    while (true) {
      ...
      int value = random.nextInt(...);
      ???.process();
    }
  }
}
```

How do we store the reference **???** to a **StarPrinter** object in the **Generator** class?

4.

```
public class Generator {
  private ??? x;

  public void run() {
    ...
    while (true) {
      ...
      this.x.process();
    }
  }
}
```

What is the type of the attribute **x**?

5.

```
public class PlusPrinter {
  public void process() {
    System.out.println("+");
  }
}
```

How can we modify the type of the attribute **x** and the classes **StarPrinter** and **PlusPrinter** so that the class **Generator** can use both?

6. How do we initialize the **listener** attribute of the **Generator** class?

7. Which changes do we have to make if we want to associate multiple listeners with the generator? For example, we would like a $*$ and $+$ to be printed whenever an integer is produced.

8. Instead of

   ```
   private Listener listener;
   ```

   what do we use to represent a collection of **Listener**s?

9. Where and how do we initialize the attribute **`listeners`**?

10. How do we add a listener to the **`listeners`**?

11. How do we invoke the **`process`** method on the **`listeners`**?

12. Whenever the **`Generator`** produces an integer, we want to print it. How does the **`Generator`** pass the produced integer to the **`Listener`**?

13.

```
public class ValuePrinter implements Listener {
  public void process() {
    ???
  }

  public void process(int value) {
    System.out.println(value);
  }
}
```

Since the class **`ValuePrinter`** implements the interface **`Listener`**, it has to provide an implementation of **`process()`** and **`process(int)`**. How to implement **`process()`**?

14.

```
public class StarPrinter implements Listener {
  public void process() {
    System.out.println("*");
  }

  public void process(int value) {
    ???
  }
}
```

Since the class **`StarPrinter`** implements the interface **`Listener`**, it has to provide an implementation of **`process()`** and **`process(int)`**. How to implement **`process(int)`**?

15. The **run** method of the **Generator** class is modified as follows.

```
final int STOP = 5;
boolean done = false;
while (!done) {
  ...
  done = random.nextInt(STOP) == 0;
}
```

Whenever the **Generator** terminates, we want to print the sum of the integers it produced. Which changes have to be made to the **Listener** interface?

16. Whenever the **Generator** terminates, we want to print the sum of the integers it produced. Which changes have to be made to the **Generator** class?

```
final int STOP = 5;
boolean done = false;
while (!done) {
  ...
  done = random.nextInt(STOP) == 0;
}
```

17. Whenever the **Generator** terminates, we want to print the sum of the integers it produced. Which changes have to be made to the **ListenerAdapter** class?

18. Implement the **SumPrinter** class?

```
public class SumPrinter          {









}
```