1 `((Bird)dodo).ability`

dodo gets cast into a `Bird` and since `Bird` gets the ability "Fly", the output of the *dodo*'s attribute ability is "Fly".

2 `dodo.ability`

The *dodo* instance receives its ability attribute from the `Dodo` class which sets the ability to "Run". Therefore, the output is "Run".

3 `dodo.getAbility()`

The method `getAbility()` in the `Dodo` class calls the `getAbility()` method of the superclass `Bird`. This is why the output is "Fly".

4 `parrot.allAbilities()`

The method `allAbilities()` of the `Parrot` class calls the `allAbilities()` method of the superclass `Bird` and adds its own ability attribute "Talk" to it. Therefore, you receive "Fly Talk" as output.

5 `parrot.ability`

The ability attribute defined in the `Parrot` class is "Talk" which is shown in the output of the *parrot* instance.

6 `carsten.ability`

The `Bird` variable *carsten* is initialized with the `Dodo` constructor. But since it is an instance of `Bird`, its ability is "Fly". This is possible because of the substitution principle.

7 `((Bird)carsten).allAbilities()`

When the `Bird` instance *carsten* is explicitly cast into a `Bird` it is still instanciated with the `Dodo` constructor which is why the output is "Run".

8 `einstein.allAbilities()`

The `Bird` *einstein* is set to the reference of *parrot* which is an instance of `Parrot`. Therefore, its abilities are "Fly" and "Talk". "Fly" from the `Bird` superclass and "Talk" from the `Parrot` class.

9 `einstein.getAbility()`

When calling `getAbility()` on the `Bird` *einstein* it only returns its `Bird` ability attribute "Fly".

10 `((Parrot)einstein).ability`

Now that the `Bird` *einstein* is cast into a `Parrot`, *einstein*.ability is overwritten to "Talk".

After that a ClassCastException is raised because `Parrot` and `Bird` are not subclasses of `Dodo`.